NPS-SP-08-002


# NAVAL POSTGRADUATE SCHOOL

**MONTEREY, CALIFORNIA**

## Simulation of the deployment and orbit operations of the NPS-SCAT CubeSat

by

Felix Roßberg

April 2008

| | |
|---|---|
| Thesis Advisor: | Dr. James H. Newman |
| Co-Advisor: | Dr. Rudolf Panholzer |
| Second Readers: | Dr.-Ing. habil. Hendrik Rothe |
| | Dr.-Ing. Klaus Krüger |

THIS PAGE INTENTIONALLY LEFT BLANK

**NAVAL POSTGRADUATE SCHOOL**
**Monterey, California 93943-5000**

Daniel T. Oliver
President

Leonard A. Ferrari
Executive Vice President
and Provost

This report was prepared for: Space Systems Academic Group.



This report was prepared by:

_____
Felix Rossberg

Reviewed by:

_____
Rudolf Panholzer, Chairman
Space Systems Academic Group

Released by:

_____
Dan C. Boger
Interim Vice President and
Dean of Research

THIS PAGE INTENTIONALLY LEFT BLANK

| REPORT DOCUMENTATION PAGE | | Form Approved OMB No. 0704-0188 |
|---|---|---|



| 1. AGENCY USE ONLY *(Leave blank)* | 2. REPORT DATE 04/24/2008 | 3. REPORT TYPE AND DATES COVERED Technical Report |
|---|---|---|

| 4. TITLE AND SUBTITLE Simulation of the deployment and orbit operations of the NPS-SCAT CubeSat | 5. FUNDING NUMBERS |
|---|---|
| 6. AUTHOR(S) Felix Roßberg, 2ndLT German Air Force | |
| 7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)<br>Naval Postgraduate School<br>Monterey, CA 93943-5000 | 8. PERFORMING ORGANIZATION REPORT NUMBER |
| 9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES)<br>N/A | 10. SPONSORING/MONITORING AGENCY REPORT NUMBER |

**11. SUPPLEMENTARY NOTES** The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

| 12a. DISTRIBUTION / AVAILABILITY STATEMENT<br>Approved for public release; distribution is unlimited | 12b. DISTRIBUTION CODE |
|---|---|

**13. ABSTRACT (maximum 200 words)**

To encourage student interest in-space education, Stanford and Cal Poly conceptualized a CubeSat in 1999. This standardized picosatellite has a cubic shape of 10 cm and weighs about one kilogram. NPS is now designing CubeSats and a structure to deploy them in-orbit as part of its emphasis on hands-on education.

This technical report deals with the simulation of the deployment and orbit operations of the NPS-SCAT CubeSat. With the help of the AGI Satellite Tool Kit software, animations are created that show the full mission of the CubeSat beginning with the launch and the deployment from the NPS CubeSat Launcher. The animations are combined into one movie, which can be used to visualize the project more easily and clearly. In addition to the animations, on-orbit systems operations such as power and communication profiles are analyzed. A MATLAB program uses the calculated data to generate the battery state of charge, so it can be determined whether the solar cells are producing enough power for the satellite. For the chosen reference mission, the results of the orbital lifetime calculation show that this CubeSat would only stay in orbit for about six months.

| 14. SUBJECT TERMS NPSCuL, CubeSat, NPS-SCAT, STK, Power budget, CubeSat Lifetime | 15. NUMBER OF PAGES 115 |
|---|---|
| | 16. PRICE CODE |

| 17. SECURITY CLASSIFICATION OF REPORT | 18. SECURITY CLASSIFICATION OF THIS PAGE | 19. SECURITY CLASSIFICATION OF ABSTRACT | 20. LIMITATION OF ABSTRACT |
|---|---|---|---|
| Unclassified | Unclassified | Unclassified | UU |

THIS PAGE INTENTIONALLY LEFT BLANK

# ABSTRACT

To encourage student interest in-space, Stanford and Cal Poly conceptualized a CubeSat in 1999. This standardized picosatellite has a cubic shape of 10 cm and weighs about one kilogram. NPS is now designing CubeSats and a structure to deploy them in-orbit as part of its emphasis on hands-on education.

This technical report deals with the simulation of the deployment and orbit operations of the NPS-SCAT CubeSat. With the help of the AGI Satellite Tool Kit software, animations are created that show the full mission of the CubeSat beginning with the launch and the deployment from the NPS CubeSat Launcher. The animations are combined into one movie, which can be used to visualize the project more easily and clearly.

In addition to the animations, on-orbit systems operations such as power and communication profiles are analyzed. A MATLAB program uses the calculated data to generate the battery state of charge, so it can be determined whether the solar cells are producing enough power for the satellite. For the chosen reference mission, the results of the orbital lifetime calculation show that this CubeSat would only stay in orbit for about six months.

THIS PAGE INTENTIONALLY LEFT BLANK

# ZUSAMMENFASSUNG

Um das Interesse von Studenten an der Luft- und Raumfahrttechnik zu wecken, entwickelten Stanford und Cal Poly 1999 den CubeSat. Dieser standardisierte Picosatellit hat eine kubische Form mit 10 cm langen Kanten und einem Gewicht von einem Kilogramm. Die Naval Postgraduate School arbeitet momentan als Teil der praktischen Ausbildung an der Entwicklung eines CubeSat und einer Trägerstruktur für den Transport ins All.

Diese Diplomarbeit beschäftigt sich mit der Simulation der Entsendung des NPS-SCAT Satelliten und dessen Betrieb im All. Dafür wird die Software Satellite Tool Kit von AGI verwendet, mit der Animationen erzeugt werden, welche beginnend mit dem Start und dem Abkoppeln den gesamten Lebenszyklus simuliert. Alle Animationen werden anschliessend zu einem Film zusammengefügt, welcher dazu genutzt werden kann, die Grundideen der CubeSat Projekte besser zu veranschaulichen.

Weiterhin wird der Betrieb im All, z.B. das Energie- bzw. das Kommunikationsprofil, untersucht. Die daraus gewonnen Daten werden in einem MATLAB Programm verarbeitet, um Aussagen über den Ladezustand der Batterien treffen zu können. Dabei wird deutlich, dass die Solarzellen ausreichend Energie erzeugen, um für eine unbegrenzte Zeit im All zu bleiben. Dagegen spricht jedoch die Berechnung der Lebensdauer des Satelliten, welche ca. ein halbes Jahr beträgt.

THIS PAGE INTENTIONALLY LEFT BLANK

## DECLARATION

Hereby I declare that this Technical Report has been created by myself using only the named sources and supplementary equipment.

## ERKLÄRUNG

Hiermit erkläre ich, dass die vorliegende Studienarbeit von mir selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel erstellt wurde.

Monterey, 24. April 2008

Felix Roßberg, Lieutenant

THIS PAGE INTENTIONALLY LEFT BLANK

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

# LIST OF ABBREVIATIONS

| | |
|---|---|
| AGI STK | Analytical Graphics, Inc. Satellite Tool Kit |
| Cal Poly | California Polytechnic State University |
| CCB | Common Core Booster |
| CG | Center of gravity |
| DEC | Dual-Engine Centaur |
| EELV | Evolved Expendable Launch Vehicle |
| EPF | Extended Payload Fairings |
| ESPA | EELV Secondary Payload Adapter |
| g | Gravitational acceleration constant at sea level on the Earth |
| GSO | Geosynchronous Orbit |
| GTO | Geosynchronous Transfer Orbit |
| HLV | Heavy Lift Vehicle |
| $I_{sp}$ | Specific impulse of the centaur stage |
| LEO | Low-Earth Orbit |
| LPF | Large Payload Fairings |
| LRB | Liquid Rocket Booster |
| LV | Launch Vehicle |
| $m_0$ | Initial mass |
| $m_{CS}$ | Mass of centaur stage |
| $m_{dry}$ | Dry mass |

| | |
|---|---|
| $m_f$ | Final mass |
| $m_{fuel}$ | Mass of Centaur stage engine propellant |
| $m_{fuel\_dv}$ | Fuel mass needed for the maneuver |
| $m_{PL}$ | Mass of the payload |
| MCS | Mission Control Sequence |
| MECO | main Engine Cutoff |
| MES | Main Engine Start |
| NPS | Naval Postgraduate School |
| NPSCuL | NPS CubeSat Launcher |
| NPS-SCAT | NPS – Solar Cell Array Tester |
| PL | Primary Payload |
| PLF | Payload Fairing |
| SEC | Single-Engine Centaur |
| SMS | Solar Cell Measurement System |
| SOC | State Of Charge |
| SPL | Secondary Payload |
| SRB | Solid Rocket Booster |
| XEPF | Extended EPF |

# ACKNOWLEDGMENTS

I would like to thank the participating professors, Professor Panholzer and Professor Rothe for supporting the student exchange between the Naval Postgraduate School and the Helmut Schmidt Universität – Universität der Bundeswehr Hamburg. Giving students the chance to write their thesis at NPS is a great opportunity to implement the engeneering fundamentals learned at the University of the Federal Armed Forces Hamburg and an excellent conclusion to that education.

The support of the whole Space Systems Academic Group was outstanding; it was a pleasure for me to be part of this community. Furthermore, I would like to thank my advisor NASA Visiting Professor Jim Newman and Research Associate Dan Sakoda, who supported me with all software problems.

THIS PAGE INTENTIONALLY LEFT BLANK

# I. INTRODUCTION

## A. STRUCTURE OF THIS TECHNICAL REPORT

This technical report is divided into four main chapters: the introduction, the theoretical foundations, the simulation of the CubeSat operations, and the conclusion.

The theoretical foundations (Chapter II) include the presentation of the two current CubeSat projects at the Naval Postgraduate School, the NPS-SCAT CubeSat and the NPSCuL. Another part of this chapter deals with the Atlas V Heavy Lift Vehicle (HLV), which is one of the possible launch vehicles for the CubeSats. The last section is about the AGI Satellite Tool Kit (STK) software. This program is used for all animations and calculations.

Chapter III describes the whole process of developing a simulation for the lifecycle of one CubeSat. This includes the creation of the model files, the settings for the animation, and the setup for creating a movie.

Finally, Chapter IV presents the conclusion and suggests some future work.

## B. OBJECTIVE OF THIS TECHNICAL REPORT

The objective of this technical report is to present the results of simulating the dynamics of deployment and orbit operations using the STK software. The full mission of the NPS-SCAT CubeSat beginning with the launch and the deployment from the NPS CubeSat Launcher is considered. Issues including power and communications profiles for a reference design mission and orbital lifetime are also be discussed. Finally, everything is combined in a movie for later presentations.

# II. THEORETICAL FOUNDATIONS

## A. NPS-SCAT

With the development of smaller satellite components the opportunity to build miniaturized satellites has increased. From now on it is possible to separate experiments and incorporate them into smaller satellites. Therefore, options for launch vehicles are increased and the cost for satellite production and development should be less than for larger satellites. Another positive aspect is the lower risk in case of a failure before the end of the satellite's mission design life because smaller satellites only carry the equipment that is needed to support a particular onboard experiment. The CubeSat (Figure 1) developed by Cal Poly and Stanford University followed this idea and combined this project with the goal to encourage student interest in-space engineering.

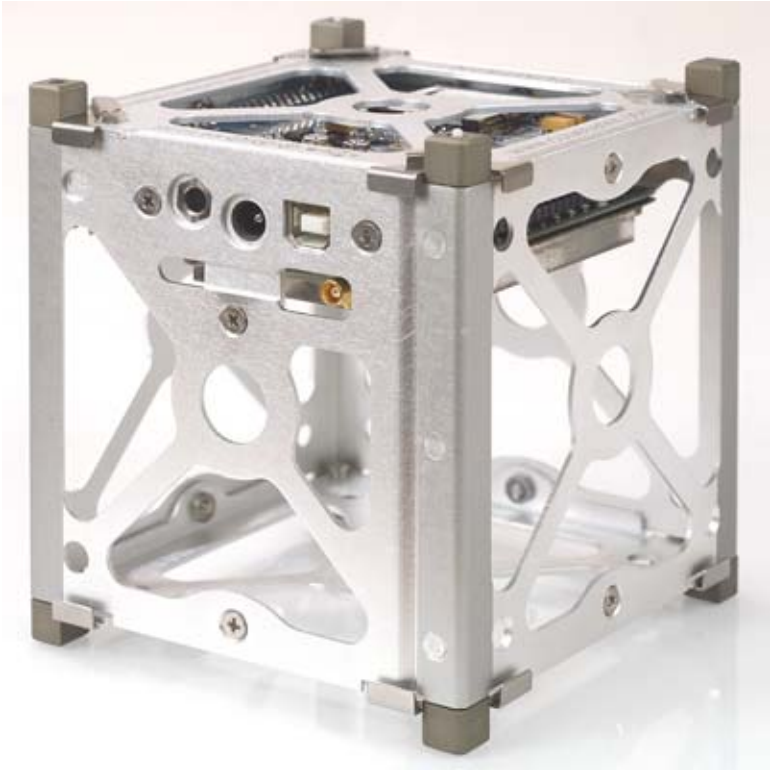


**Figure 1: 1U CubeSat structure**

This picosatellite has 10 cm long edges and a maximum weight of one kg. The standardized 1U CubeSat can already be expanded to a 3U form factor and 5U or even 2U by 2U by 3U CubeSats may be standard options in the near future. The larger configurations may be necessary to carry cameras or other video systems that have lenses with a bigger diameter than 10 cm.

As part of the hands-on education, Naval Postgraduate School (NPS) students are working on different CubeSat projects. One of them is the development of the NPS-SCAT (NPS – Solar Cell Array Tester) CubeSat, which will carry the Solar Cell Measurement System (SMS). This system will measure the in-orbit performance of experimental triple-junction cells (TJC). This project will be the first CubeSat developed by the NPS and will be used as a technology test-bed with a relatively simple design.

### B. NPSCUL

The current launch vehicles (LV) can typically carry a greater mass of payload into space than just the primary payload. A secondary payload adapter such as the ESPA-ring can accommodate a primary satellite up to 15,000 lbs. and also six secondary satellites, up to 400 lbs. each. These are available to launch on Delta IV or Atlas V. The standard secondary interface is a 15-inch diameter bolt circle with 24 fasteners.

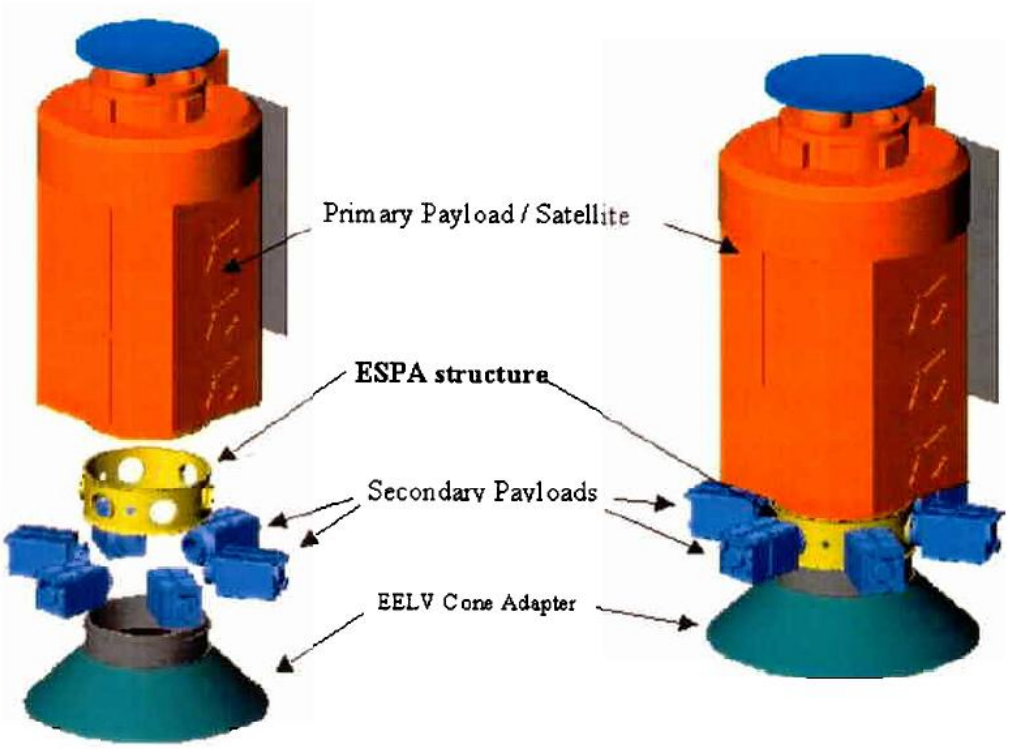



**Figure 2: ESPA on EELV**

One of these secondary payloads(SPLs) is the NPSCuL, which fits into the given ESPA envelope of 90.1 x 71.1 x 60.9 cm and is lighter than the maximum allowed mass of 181 kg. The results of a previous technical report [2] showed two different design options that can be modified for the required mission. In both cases, ten P-PODs are assembled and can be opened in a specified order. The whole structure can be covered by an outer box or can be fully enclosed using an additional lid.

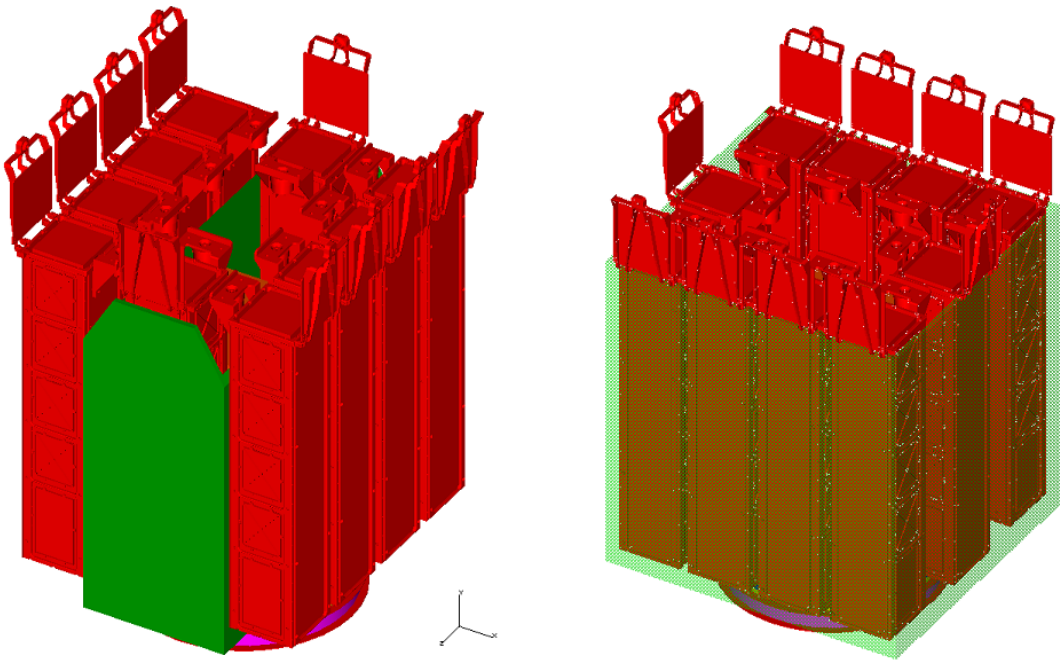


**Figure 3: Open D-advanced structure and Box-structure [2]**

### C. ATLAS V HEAVY LIFT VEHICLE

The Atlas V 400 and 500 series launch vehicles are the latest versions of the Atlas launch system and were placed into service in 2002. Since 2007 the Atlas V series is distributed by the Lockheed Martin - Boeing joint venture United Launch Alliance. The main part of Atlas V is the Common Core Booster (CCB), which has an RD-180 engine with a thrust of 4,152 kN. Up to five strap-on Solid Rocket Boosters (SRB), a Centaur in either the Single-Engine Centaur (SEC) or the Dual-Engine Centaur (DEC) configuration, and one of several Payload Fairings (PLF) are the other components of the launch vehicle. A three-digit (XYZ) naming convention was developed for the Atlas V 400 and 500 series to identify its multiple configuration possibilities.

1st digit – payload fairing diameter (4 or 5 meter)

2nd digit – number of solid rocket booster (between zero and 5)

3rd digit – number of centaur engines

For example, the Atlas V 541 has a Payload Faring with a 5 m diameter, four additional CCBs and a SEC.

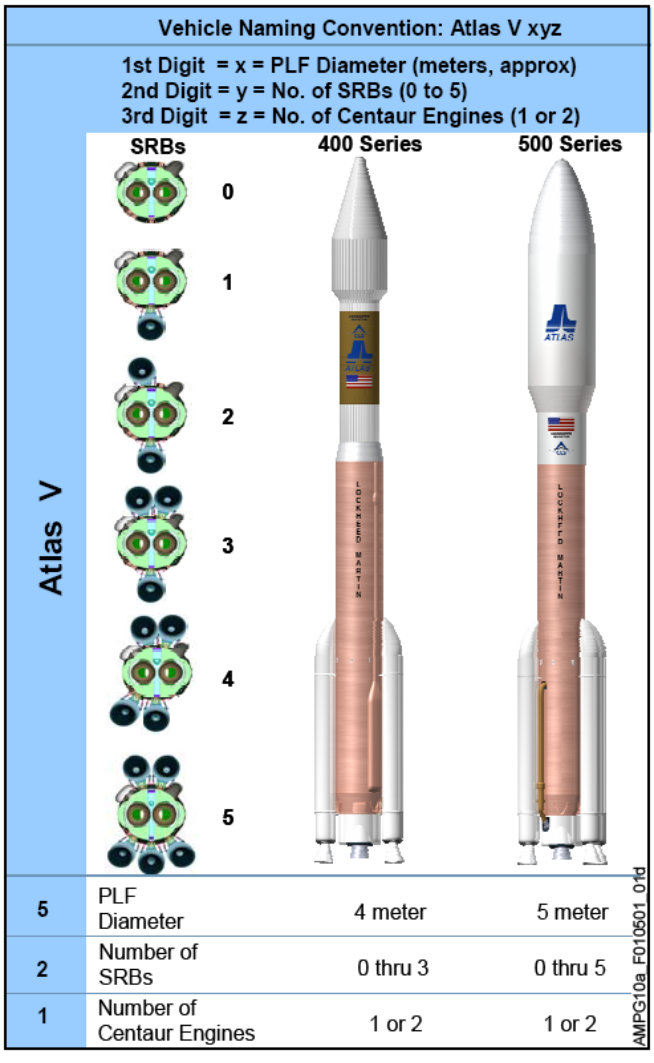



**Figure 4: Atlas V Naming Designator Definition [4]**

Flight-proven and operational beginning in August 2002, the Atlas V is meeting a wide variety of commercial and U.S. Government launch requirements today. The Atlas V family includes the flight-proven Atlas V 400 and 500 series and the Atlas V HLV, which is in development. Until now, 13 launches took place and the success rate is 92% [8].

The Atlas V 400 series incorporates the flight proven 4 m diameter Atlas V 12.0 m Large Payload Fairing (LPF), the 12.9 m Extended Payload Fairing (EPF), or the 13.8 m Extended EPF (XEPF).

The Atlas V 500 series incorporates the flight proven 5 m diameter 20.7 m short, the 23.5 m medium, or the 26.5 m long payload fairing. The Atlas V Heavy Lift Vehicle (HLV) configuration is currently under development and incorporates the three 5 m payload fairings.

The PLF encloses and protects the spacecraft during ground operations and launch vehicle ascent. The PLF also incorporates hardware to control thermal, acoustic, electromagnetic, and cleanliness environments for the spacecraft and may be tailored to provide access and radio frequency communications to the encapsulated spacecraft [4].

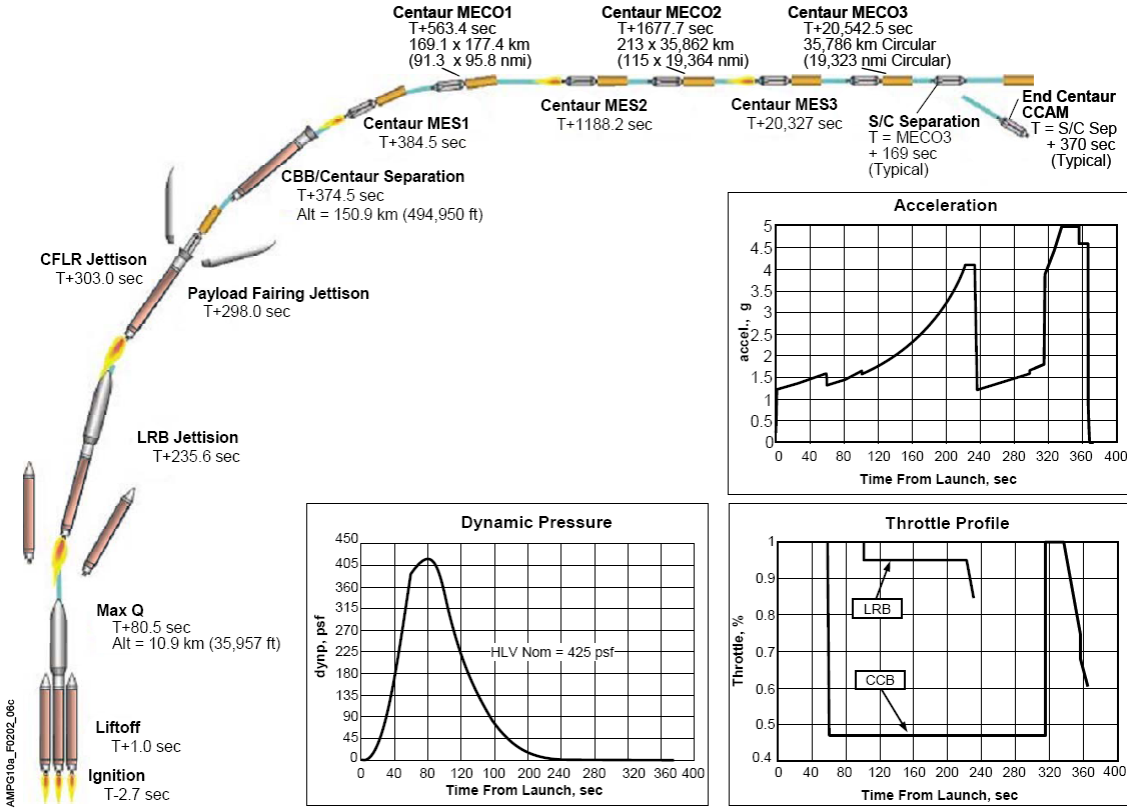
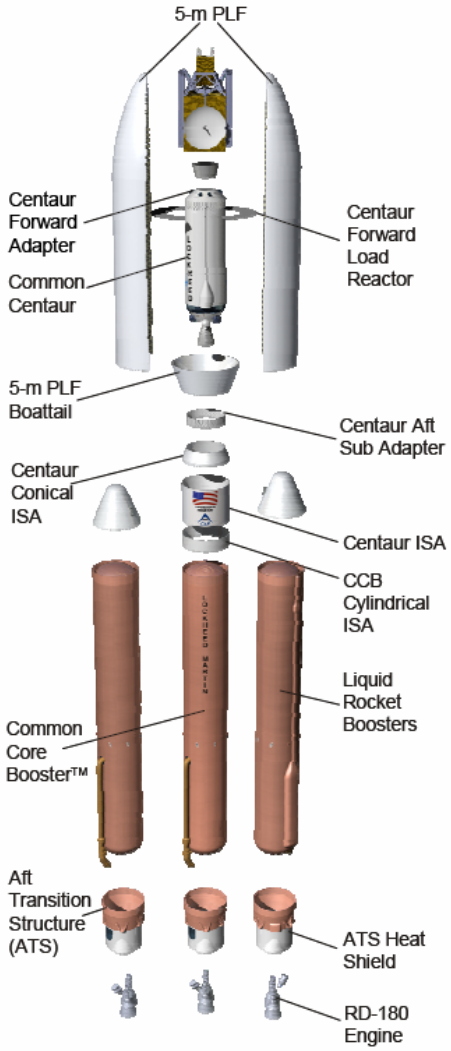



**Figure 5: Typical Atlas HLV GSO Ascent Profile**

A special version is the Atlas V Heavy or HLV, which consists of the main CCB and has two additional SRB that are the same as the main CCB. The centaur stage can be a SEC, for example Atlas V HLV 5H1, for bringing satellites to geosynchronous transfer orbit (GTO). Alternatively, a DEC (Atlas V HLV 5H2) can carry the payload to low-earth orbit (LEO). The possible payload is between 13,605 kg (GTO) or 25,000 kg (LEO).




**PAYLOAD FAIRING**

| | |
|---|---|
| **Features** | **5-m Long** |
| Diameter: | 5.4 m |
| Length: | 26.5 m |
| Mass: | 4,394 kg |
| **Subsystems** | |
| Fairing: | Bisector; Sandwich Construction with Graphite Epoxy Face Sheets & an Aluminum Honeycomb Core |
| Boattail: | Fixed, Composite Sandwich Construction |
| Separation: | Vertical Separation by a Linear Piston & Cylinder Activated by a Pyrotechnic Cord; Horizontal Separation by an Expanding Tube Shearing a Notched Frame, Activated by a Pyrotechnic Cord |

**COMMON CENTAUR**

| | |
|---|---|
| **Features** | |
| Size: | 3.05-m dia x 12.68-m Length with Extended Nozzle |
| Inert Mass: | 2,316 kg |
| Propellant: | 20,860-kg $LH_2$ & $LO_2$ |
| Guidance: | Inertial |
| **Subsystems** | |
| Structure: | Pressure Stabilized Stainless Steel Tanks Separated by Common Ellipsoidal Bulkhead |
| Propulsion: | One or Two Pratt & Whitney Restartable Engine(s) |
| — Model: | RL 10A-4-2 |
| — Thrust: | 99.2 kN (SEC) 198.4 kN (DEC) |
| — $I_{SP}$: | 450.5 s |
| (SEC) | One Electromechanically Actuated 51-cm Columbium Fixed Nozzle<br>Four 27-N Hydrazine Thrusters<br>Eight 40.5-N Lateral Hydrazine Thrusters |
| (DEC) | Two Hydraulically Actuated 51-cm Columbium Extendible Nozzles<br>Eight 40-N Lateral Hydrazine Thrusters<br>Four 27-N Hydrazine Thrusters |
| Pneumatics: | Common with Atlas V 400/500 Series |
| Avionics: | Common with Atlas V 400/500 Series |
| Insulation: | Polyvinyl Chloride Foam (1.6-cm Thick), Modified Adhesive Bonding with Optional Radiation Shields |

**LIQUID ROCKET BOOSTERS (LRB)**

| | |
|---|---|
| **Features** | |
| Size: | 3.81-m dia x 35.99-m Length |
| Propellant: | 286,795-kg $LO_2$ & RP-1 (Each) |
| Inert Mass: | 21,902 kg |
| Thrust: | 4,152-kN VAC 100% Thrust (Each) |
| $I_{SP}$: | 338.4 s |
| Structure: | Same as Common Core Booster |

**CENTAUR INTERSTAGE ADAPTER (C-ISA LARGE)**

| | |
|---|---|
| **Features** | |
| Size: | 3.81-m dia x 4.46-m Length |
| Mass: | 2,292 kg (Includes ISA, Aft Stub Adapter and Boattail) |
| **Subsystems** | |
| Structure: | Composite Sandwich (Aluminum Core/Graphite Epoxy Face Sheets) |

**CCB CYLINDRICAL INTERSTAGE ADAPTER**

| | |
|---|---|
| **Features** | |
| Size: | 3.81-m dia x 0.60-m Length |
| Mass: | 282 kg |
| **Subsystems** | |
| Structure: | Aluminum Machined Rolled-Ring Forging |

**COMMON CORE BOOSTER™ (CCB)**

| | |
|---|---|
| **Features** | |
| Size: | 3.81-m dia x 32.46-m Length |
| Propellant: | 286,795-kg $LO_2$ & RP-1 |
| Inert Mass: | 23,648 kg |
| Guidance: | From Upper Stage |
| **Subsystems** | |
| Structure: | Structurally Stable Aluminum Isogrid Tanks; Integrally Machined Aft Transition Structure; Composite Heat Shield |
| Separation: | 8 Retro Rockets |
| Propulsion: | Pratt & Whitney/NPO Energomash RD-180 Booster Engine (2 Chambers)<br>SL 100% Thrust = 3,827 kN, $I_{SP}$ = 311.9 s<br>Vac 100% Thrust = 4,152 kN, $I_{SP}$ = 338.4 s |
| Pneumatics: | Helium (Pressure Tanks), Computer-Controlled Pressurization System |
| Hydraulics: | Integral with Engine Provides Gimbal Control |
| Avionics: | Flight Control, Flight Termination, Telemetry, Redundant Rate Gyros, Electrical Power |

AMPG10a_F010501_05c

**Figure 6: Atlas V HLV Launch System [4]**

## D. AGI SATELLITE TOOL KIT

AGI software technology is a time-dynamic, physics-based geometry engine that answers fundamental questions essential to solving all dynamic analysis problems. Independently validated and verified, AGI analysis software is used as a stand-alone, out-of-the-box desktop application or integrated into networked, enterprise, or service-oriented, composite applications or real-time environments. As a commercially available solution, AGI software gives users immediate access to proven technology and the flexibility to customize their solutions.

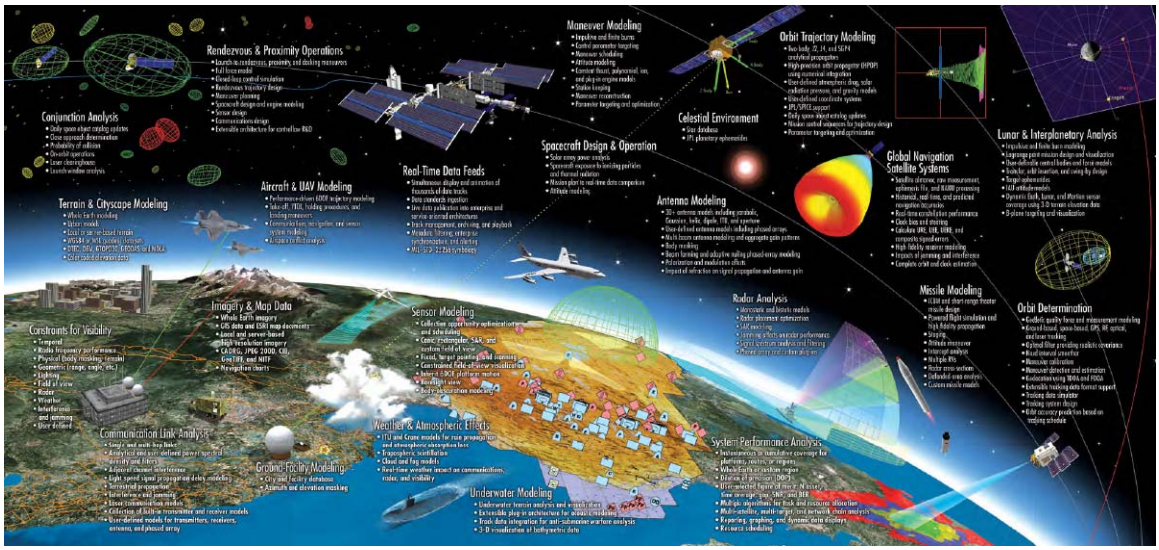



**Figure 7: Possible AGI applications [5]**

STK performs complex analysis of land, sea, air, and space assets, and shares results in one integrated solution. STK functionality enables users to:

- Calculate position and orientation
- Evaluate inter-visibility times
- Determine quality of dynamic spatial relationships

Add-on modules are available to extend STK functionality for communication and radar analyses; maneuver planning; collision avoidance; cumulative coverage analyses; high resolution imagery and terrain; parametric analysis and optimization; complex mission scheduling; missile systems design, flight, and interception; software integration and customization; and more [5].

For creating this paper the AGI Satellite Tool Kit, Version 8.1 was used. Furthermore, the provided tool Modeler 8 was used to display and review the created model files and articulations.

THIS PAGE INTENTIONALLY LEFT BLANK

# III. SIMULATION OF THE CUBESAT OPERATIONS

## A. MODEL FILES

### 1. How to create a model file

STK offers a lot of different space, air, sea, and land model files. Nevertheless, in case of working with an own developed spacecraft you have to create a model file to integrate it into the scenario. This can be done with a usual text editor and the results can be checked with the provided AGI Modeler 8.

A model is a graphical representation of an object. It is defined in a model file, which is an ASCII text file with an `.mdl` extension. As shown in Figure 8 the model file is arranged in a hierarchy of primitives and components. A component can contain primitives that define the shape of the component (such as a polygon or a cylinder), parameters that describe it in some way (such as color or shininess), or primitives that refer to other components. In the hierarchy, the component is the parent off all primitives and subcomponents contained within. A model file contains the following:

- Description – A detailed description of the model
- Components – Each component describes a specific part of the model
- Primitives – A component contains one or more primitives that define the shape of the component
- Parameters – Parameters are used to further describe a primitive or component

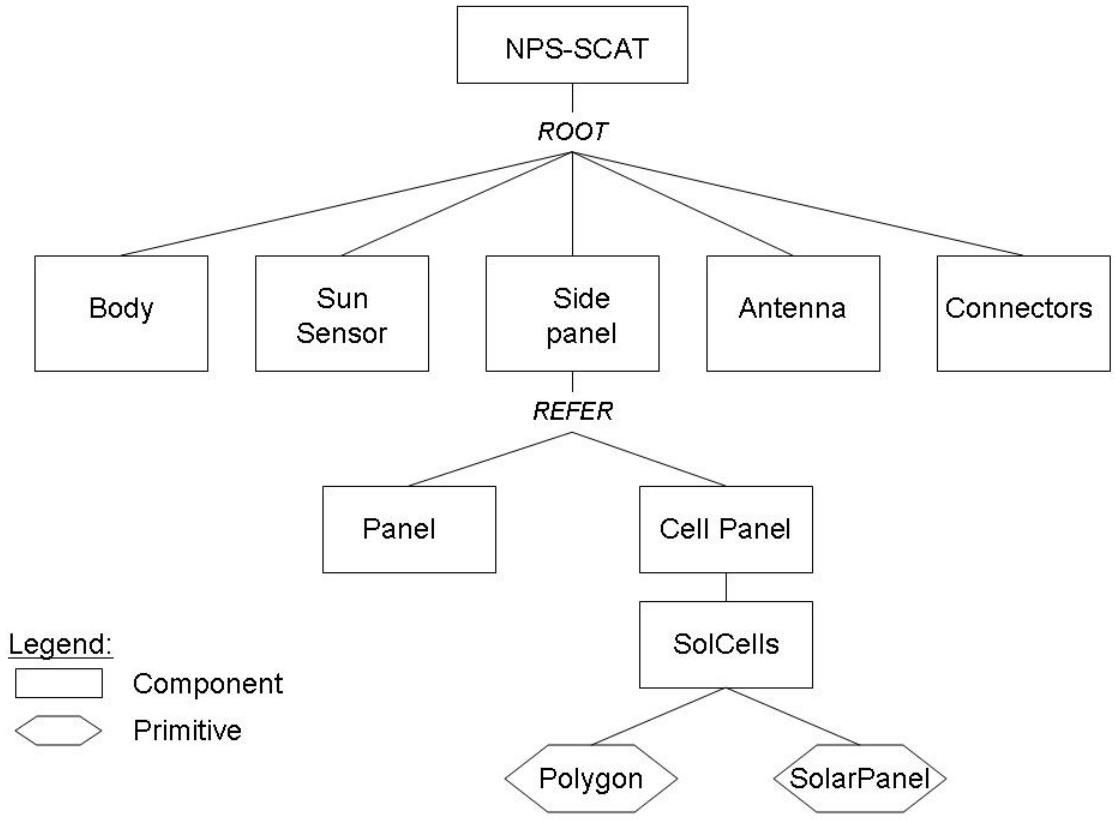



**Figure 8: Model file hierarchy**

A model file is not necessarily ordered sequentially from the beginning of the file. The `Root` designator, which must be contained in one and only one component, signifies the top of the hierarchy for the model. The hierarchy begins with the component specified as `Root` and traverses down the hierarchy, building each of the components included in the model file.

A component contains one or more primitives that define the shape of the component. A primitive specifies a geometric construct, such as a cylinder or a sphere or refers to another component in the model. Primitives can be modified by parameters such as color, style and translucency. Most primitives include a data section that specifies the coordinates of the primitive.

Transformations are parameters that statically define the position and orientation of a component or primitive, such as the location of an antenna on the body of a satellite. Transformations do not change the way an STK object is defined or affect access or other calculations between objects being modeled. They can be used effectively to visually resolve anomalies or to provide realistic motion in the models for animation.

There are three ways to transform a component:

- Rotate – Spins a component around the x, y and/or z axes of the parent component
- Scale – Expands or contracts a component relative to the x, y and z axes of the parent component
- Translate – Moves a component relative to the x, y or z axes of the parent component.

All children inherit the transformations of the parent component. All transformations occur in the order in which they appear within a component.

The units in a model file are specified in meters. Models can be built in any unit, but the `Root` component in the file assumes they are in meters. [1]

## 2. NPS-SCAT model file

Basically, a four-sided cylinder with end faces describes the shape of the NPS-SCAT CubeSat. There was also added a movable antenna and eight spacers. In the following, the structure of the model file `NPS-SCAT.mdl` will be described. Solar panels are also defined to take advantage of STK's solar power analysis tools.

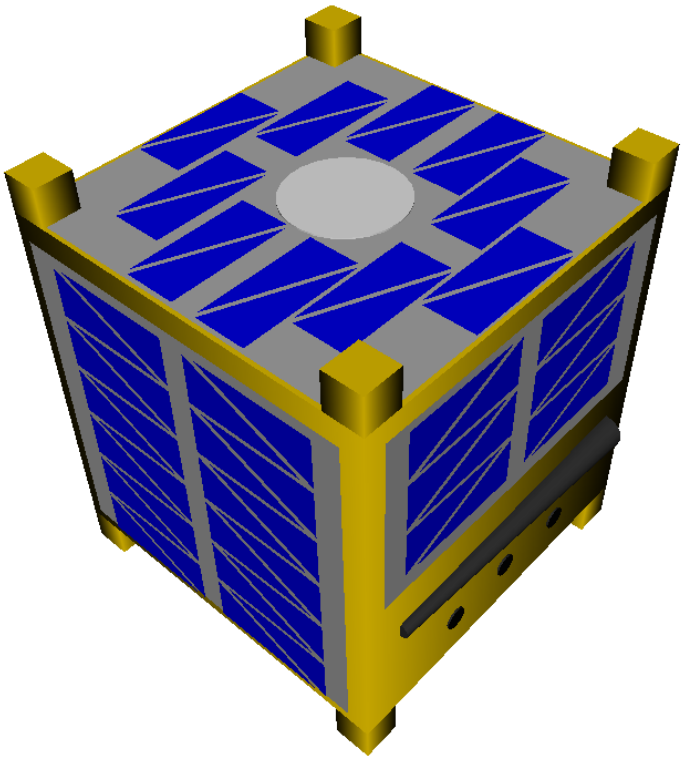


**Figure 9: NPS-SCAT model**

Line 1 – 11: Description

Line 12 – 17: Cell efficiency

The keyword `SolarPanelGroup` specifies the name of a group of solar panels and their efficiency. This is needed to calculate values in STK with the satellite tool `Solar Panel`. The efficiency rating depends on the type of panels used. The higher the number, the more efficient the panels. This

number is a percentage of how well the cell can convert solar to electrical energy.

SolarPanelGroup <groupname> <efficiency>

Line 18 – 35: Mainstructure

The main structure of the NPS-SCAT CubeSat is a four-sided cylinder. The Cylinder primitive causes a cylinder to be drawn, where NumSides are the number of sides, Face1Radius is the radius of one end of the cylinder in meters, Face1Normal is the normal direction of the cylinder based on X, Y and Z coordinates (magnitude of normal must be greater than 0.0). The same definitions are used for Face2Radius and Face2Normal, which is the opposite side of the cylinder. The Length command defines the longest extent of a component, measured from end to end in meters.

```
Component Body
     Rotate 45 0 0
     Translate 0 0.05 0.05
     Cylinder
          FaceColor gold
          NumSides 4
          Face1Radius 0.07
          Face1Normal -1 0 0
          Face2Radius 0.07
          Face2Normal 1 0 0
          Length 0.1
     EndCylinder
EndComponent
```

Line 36 – 51: SunSensor

The component SunSensor creates the sensor, which will be mounted on top of the CubeSat and measure the sun angle.

Line 52 – 66: Antenna

Definition of the Antenna by using the Cylinder primitive.

Line 67 – 81: Connectors

Definition of the Connector by using the Cylinder primitive.

Line 82 – 121: Panels for SolarCells

Definition of the Panels for the SolarCells by using the Polygon primitive. Panel is used for every side, except the side with the sun sensor (Panel2) and the side with the connectors and antenna (Panel3).

Line 122 – 137: SolarCells

The component SolCells defines the solar cells that are used on NPS-SCAT. The keyword SolarPanel provides access to the solar panel tool. This parameter identifies a component or primitive as a solar panel and identifies the group to which the solar panel belongs. If you specify a SolarPanel, you must also specify a SolarPanelGroup. If there is only one SolarPanel, the solar panel group name is the same name as the solar panel. The triangular shape of the cells is defined by the Polygon primitive.

```
Component SolCells
     SolarPanel npsscat
     Polygon
          FaceColor blue
          Numverts 3
          Data
          0 0 0
          0 0.0315 0
          0 0 0.0132
     EndPolygon
EndComponent
```

Line 138 – 153: Cellpair

A Refer primitive points to a component. In this case, the Cellpair component consists of two solar cells (SolCells). The Rotate command turns the component about the X-, Y- or Z-axis. Rotations are applied in consecutive order, beginning with the X-axis (i.e., X, and then Y, and then Z). Two or more Rotate commands can be used to rotate a component out of the <rx>, <ry>, <rz> sequence. The transformation Translate moves an object along the X-, Y- or Z-axis.

```
Component Cellpair
     Refer
     Translate 0.001 0 0
     Component SolCells
     EndRefer
     Refer
     Rotate 180 0 0
     Translate 0.001 0.0325 0.014
     Component SolCells
     EndRefer
EndComponent
```

Line 154 – 168: Experimental Cells

The onboard experiment will trace current-voltage (I-V) curves of the triple-junction cells while in orbit. The generated power is not used for the power budget of the CubeSat.

The triple-junction cells have the same shape as the other solar cells. That is why the `Polygon` primitive is used again to create triangular cells.

Line 169 – 184: TJC Cellpair

Two triple-junction cells are combined to a cell pair (`TJC_Cellpair`).

Line 185 – 209: String assembly

In these lines, a solar cell string (`Cellpanel`) is created out of four solar cells by referring, translating, and rotating single `SolCells`.

Line 210 – 243: Assembly SolarCell sidepanel

The component `Sidepanel` consists of six `Cellpanels` that are placed on one Panel. To move each cell panel into the right position, transformation commands are used.

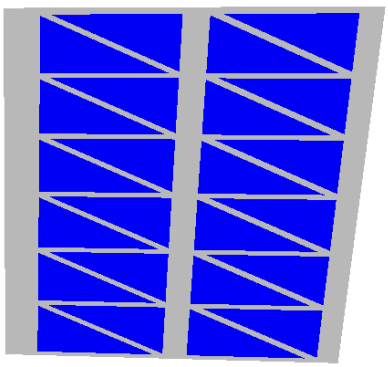


**Figure 10: NPS-SCAT SidePanel**

Line 244 – 298: Top panel

The `Top` consists of the `Bottom` background, `Panel3`, eight `Cellpairs`, two `TJC-Cellpairs`, and the `SunSensor`.

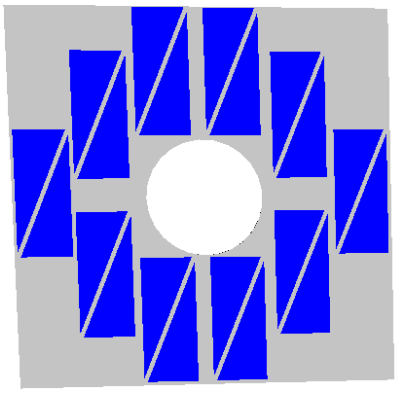


**Figure 11: NPS-SCAT TopPanel**

Line 299 – 353: Bottom panel

The `BottomPanel` is equal to the Top, but there are no triple-junction cells that are used for the SMS.

Line 354 – 391: AntennaPanel

The `AntennaPanel` consists of three `CellPanels` (corresponding to six solar cells), three `Connectors` and `Panel2` as a background.

Line 392 – 407: Spacer

The spacers are the small cubes on top and on the bottom of the CubeSat, which carry the switches for the separation. They are described by an extrusion (`Spacer`) and a covering polygon (`SpacerPanel`).

Line 408 – 421: Articulation Antenna

The component `Deployable1` describes the articulation of the movable antenna. Articulations in a model file are dynamic transformations that

can be applied to primitives or where <Articulation Name> uniquely identifies the articulation from all others in the file. These commands require a separate command for each axis to be transformed. Articulation commands define the range of movement allowed for the articulation itself. An Articulation command is formatted as:

<Articulation Command> <Articulation Command Name> <Min Value> <Init Value> <Max Value>.

Preferred articulation commands are the translation, rotation, and scaling about a certain axis. The articulation in this file is called Antenna_Deployable and consists of a Rotation about the Y-axis. The minimum value is -90 degrees, the init value is 0 degree and the maximum value of the rotation is 0 degree as well.

After defining the articulation, it has to be applied to the movable component. This is done with the help of the Refer command.

```
Component Deployable1
        Articulation Antenna_Deployable
                yRotate Rotate -90.  0 0
        EndArticulation
        Refer
                Component Antenna
        EndRefer
EndComponent
```

Line 422 – 504: NPS-SCAT ASSEMBLY

This block contains as shown in Figure 8 the Root command, because this is the final assembly. The origin of the coordinate system is moved and rotated before, so that it has the same place and orientation as the CG of the CubeSat. Every previous defined component is assembled with the Refer command to obtain the final shape of NPS-SCAT.

### 3. CubeSat launch vehicle model file

After creating the NPS-SCAT model file the satellite and the launch vehicle, which consists of the Atlas V HLV and the supporting structure NPSCuL, has to be assembled. Therefore, the `atlas-v-heavy.mdl` file given by STK (`C: → Program Files → AGI → STK 8 → STKData → VO→ Models → Space`) was modified and expanded. The content of the new created file (`atlas-v-heavy-NPS-SCAT.mdl`) will be described in the following.

Line 1 – 24311: Atlas V heavy model file

Nothing was changed in the file that was created by STK in 2002. The only modification was the erasure of the `Root` command, because otherwise all the changes had to be integrated to the `atlas5heavy_full_ROOT` component.

**Figure 12: Atlas V heavy model**

Line 24312 – 24326:

This is the description of the file.

Line 24327 – 24488: D-structure

The results of a previous technical report showed, that only two different structure options for an NPSCuL are possible. In this case, the open D-advanced structure is used as the supporting structure [2]. Therefore, the different parts (`FrontPart`, `BottomPart`, `SidePart`) are created in lines 24334 – 24463. The whole structure is assembled between the lines 24464 and 24488 (`D-advanced`).

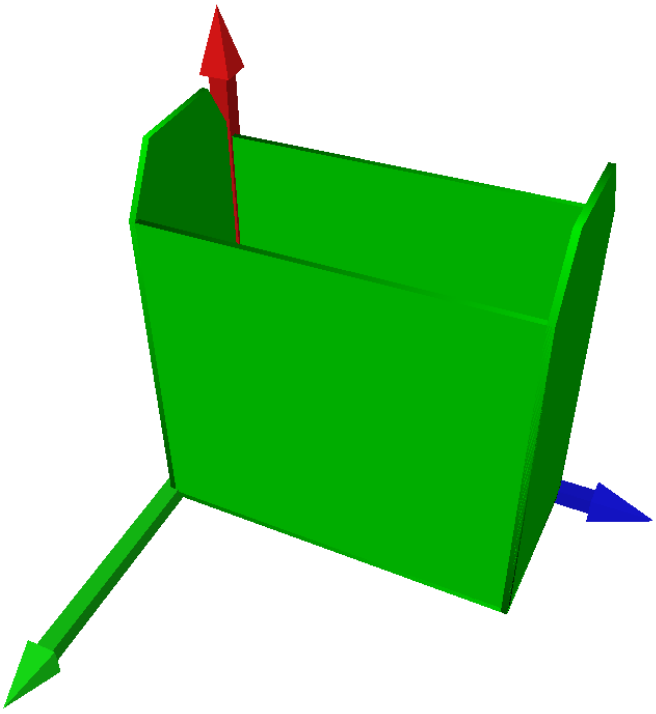


**Figure 13: Open D-advanced structure**

Lines 24489 – 24974: P-POD MK III

The P-POD MK III is the standardized mechanical and electrical connection between the CubeSats and the launch vehicle. Actually, CalPoly only offers a 3U P-POD, but extended versions are under development. In this case, the 5U MK III P-POD is used.

Lines 24497 – 24525: `P-cover`

The `P-cover` is the basic shape (long box) of the P-POD.

Lines 24526 – 24570: `Bracket`

The `Bracket` describes the joint between the lid and the P-cover.

Lines 24571 – 26636: `Opening Mechanism`

The `Opening Mechanism` is the part of the structure, which is added to the front side of the P-Pod. It includes the supporting structure and the non-pyrotechnic mechanism itself.

Lines 24572 – 24768: `Lid`

The `Lid` is the covering panel at the top of the P-POD. This part is movable and that is why an articulation has to be defined. If only one articulation would be defined, it would only be possible to open all lids at the same time. Therefore, 10 articulations (`Deployable1` – `Deployable10`) are created.

Lines 24769 – 24974: `P-POD Assembly`

The P-POD Assembly is joining all parts of the P-POD together. This is done for all 10 P-PODs (P-POD1 - P-POD10). Another possible way to create the P-POD model would be to convert 3D-CAD models to STK `*.mdl` files with the help of a converting tool. This would result in a long model file that consists of complex matrixes where the user would not be able to modify anything, for example the needed movable lid.

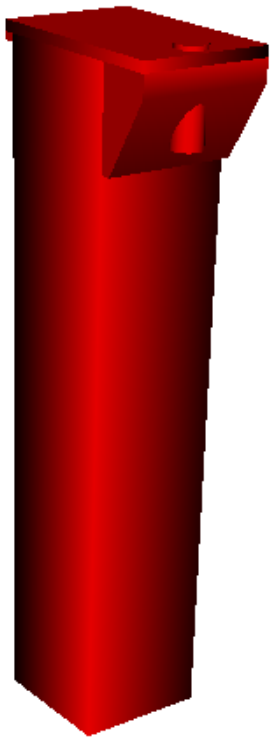


**Figure 14: P-POD model**

Lines 24975 – 25036: Base

This part consists of the `BasePlate` and the upper ring of the lightband (`UpperLightBand`). The baseplate has a square bolt pattern for the connection to the D-advanced structure and a circle bolt pattern on the other side for the connection to deployment mechanism (if required) or the ESPA-ring. In this case, the Mark II Lightband of the Planetary Systems Corporation will be used.

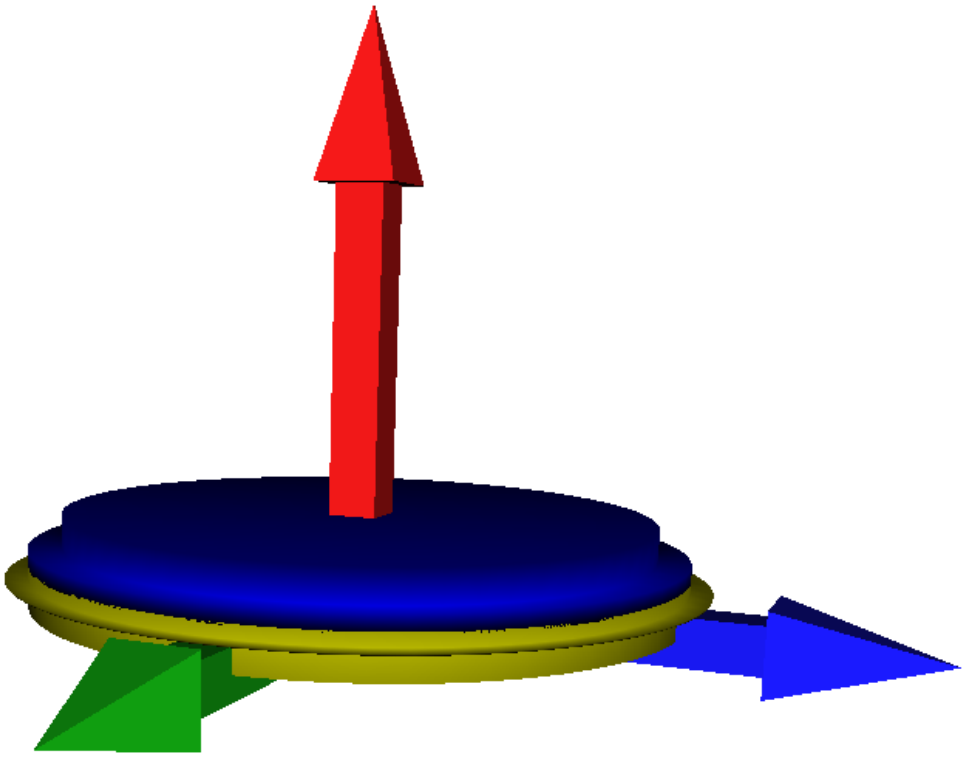


**Figure 15: Baseplate and upper ring of lightband**

Lines 25038 – 25080: Electronics Box

Another part of the NPSCuL is the `ElectronicsBox`, which is designated to carry all of the necessary electronic equipment and batteries. This box can also be used to store additional mass for getting a certain CG or weight.

Lines 25081 – 24248: ESPA-ring and Secondary Payloads

As shown in Figure 2 the ESPA-ring is a part that assembles up to six SPLs radially. The basic shape is created between the lines 25091 and 25102 (`EspaCover`) and the six connectors are defined between lines 25103 and 25117 (`EspaConnector`). The open spot in Figure 16 is the place where the NPSCuL will be added. The orange ring in this picture is the lower lightband (`LowerLightband`) which stays with the ESPA-ring after the separation of the SPL.

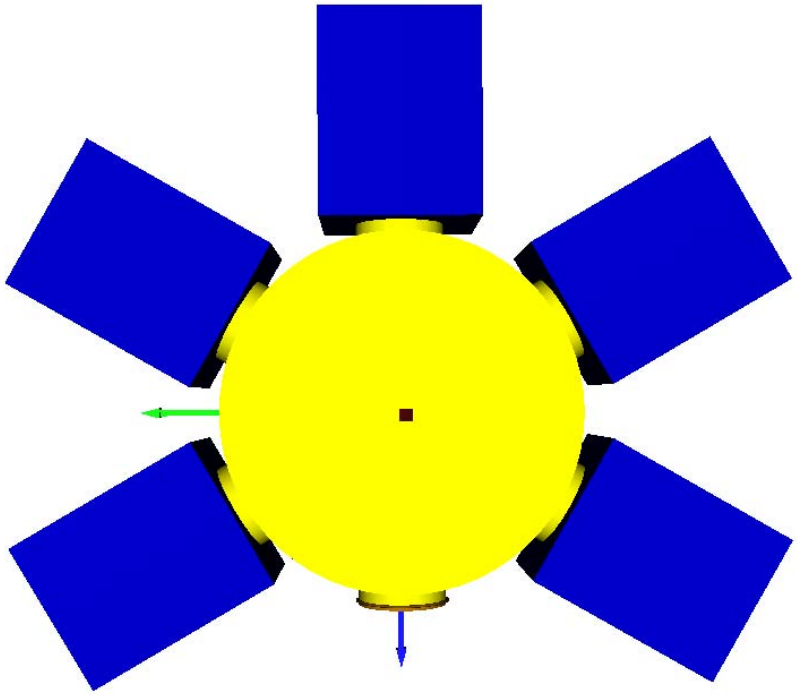


**Figure 16: ESPA-ring and SPLs**

Lines 25247 – 25318: NPSCuL-structure Assembly

This part deals with the assembly of the `NPSCuL-structure`. As shown in Figure 17 the origin of the coordinate system is situated in the middle of the d-advanced structure. Therefore, all other parts have to be translated to their final position. The used dimensions are the results of the design developments in [2].

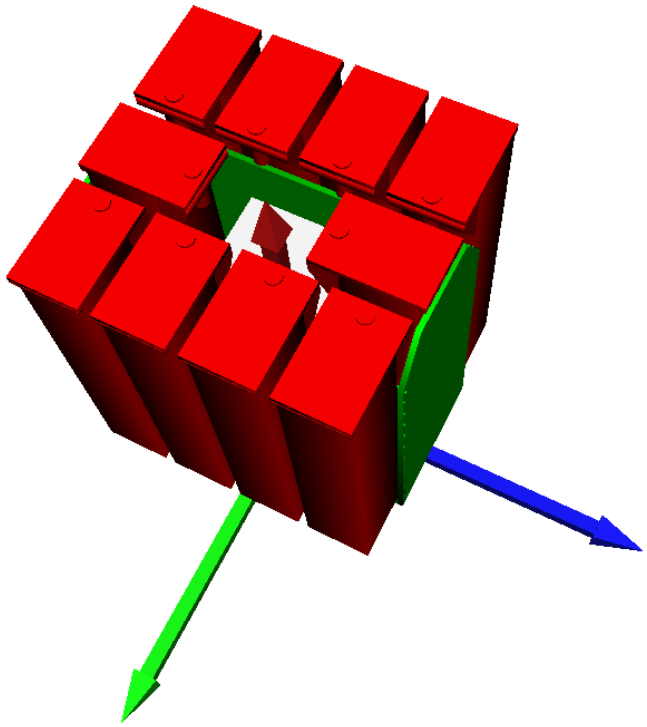


**Figure 17: NPSCuL model**

Lines 25320 – 25732: Primary Payload

The main idea of small satellites is to use the remainder of mass not needed by the primary payload. In this simulation, NPSAT1 will be used as the primary payload. This satellite is also a small satellite, which is developed by the Space Systems Academic Group at the NPS. To make the simulation more realistic, the `npsat1` model created by Research Associate Dan Sakoda in 2004 will be scaled to reach the dimensions of a typical primary payload. A cone (`PPLCone`) describes the connection between the primary payload and the ESPA-ring.

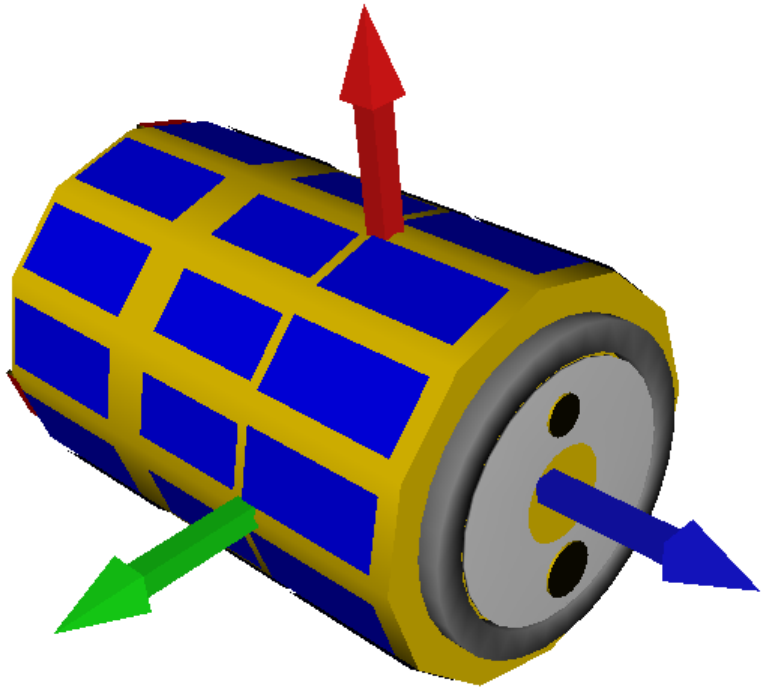


**Figure 18: NPSAT1 model**

Lines 25732 – 26233: NPS-SCAT

The NPS-SCAT CubeSat is also part of this model file, the detailed description of this component can be found under III.A.2.

Lines 26234 – 26350: Scenario Assembly

This part of the model file is the final assembly of all components created before. The component `LV` assembles the Atlas V launch vehicle (`atlas5heavy_full_ROOT`) with the connecting cones (`PPLCone`), `NPSAT1`, and the `ESPA-ring`. Afterwards, the articulation `Deployable11` for the deployment of NPSCUL is defined. Therefore, the launch vehicle will be dropped in Z-direction (minimum value: -100, start value 0, maximum value 0). The movable launch vehicle and NPSCuL are assembled in `ScenarioLV`. The final step of the launch will be the deployment of NPS-

SCAT, which is created in `Deployable12` (separation in X-direction and scaling about the Y- and Z-axis). The Articulation defined between the lines 26274 and 26282 (`Deployable14`) is used to scale the CubeSat. This is necessary for the next steps of the simulation and will be described in III.D. `ScenarioCubeSat` is the last component of the model file and assembles the articulations created before.

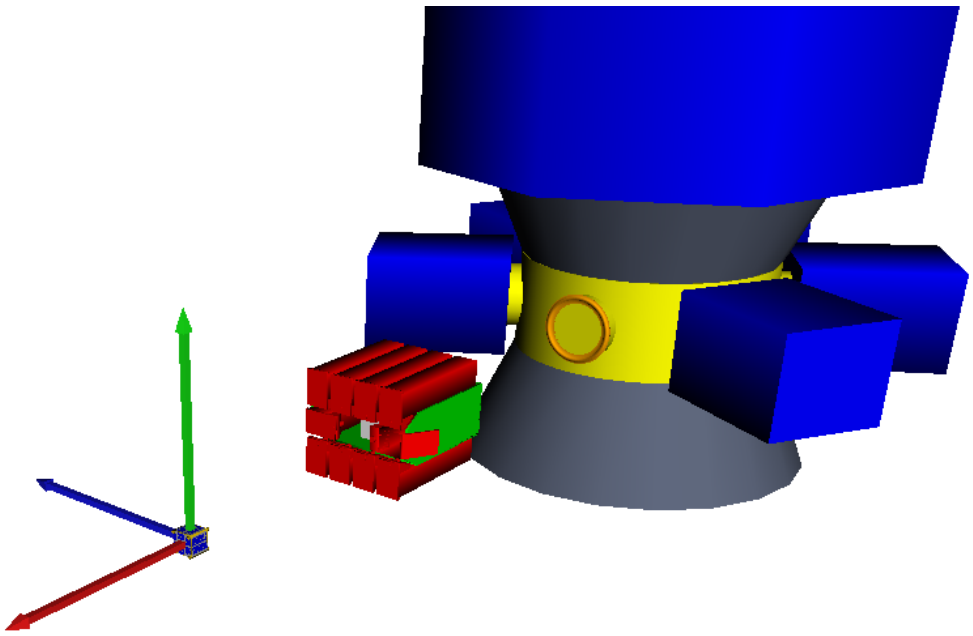


**Figure 19: Deployment of NPS-SCAT**

### 4. Atlas-Launch pad model file

The simulation will start with the launch of the Atlas V rocket at Cape Canaveral. Therefore, the Atlas launch pad provided by STK is used (`C: → Program Files → AGI → STK 8 → STKData → VO→ Models → Land → atlas-lp.mdl`). The only thing that was changed in this model file is the articulation `ServTwr_Rollback`, which is used to move the service tower away from the launch pad. The modified model file (`atlas-lp-modified.mdl`) has zero as the minimum value, 60 as the start value, and 60 as the maximum value.

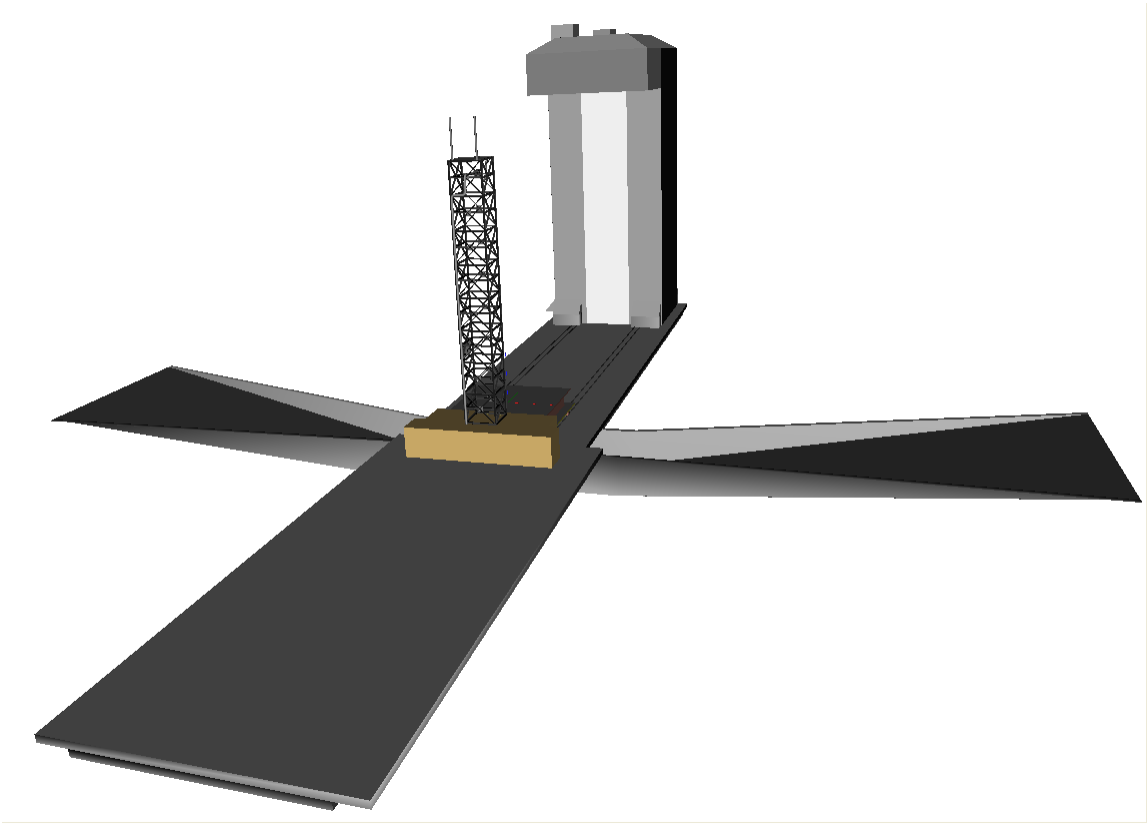


**Figure 20: Atlas V launch pad**

## B. STORY BOARD

The timeline for the launch is given by the Atlas V sequence of events (Table 1). In this case, the primary payload is designated for the geosynchronous orbit (GSO). The GSO is a direct, circular, low-inclination-orbit around Earth having a period of 23 hours 56 minutes 4 seconds and a corresponding altitude of 35,784 km (22,240 miles, or 6.6 Earth radii). In such an orbit, a satellite maintains a position above Earth that has the same longitude. However, if the orbit's inclination is not exactly zero, the satellite's ground-track describes a figure eight [3].

**Table 1: Typical Atlas V Launch Vehicle Mission Sequence Timeline [4]**

| Atlas V Mission Sequence | 401 Standard GTO | 401 LEO Sun-synch | 521 Standard GTO | 521 Extended Coast GTO | 521 3-Burn GSO | HLV GSO |
|---|---|---|---|---|---|---|
| (Time from T-0) | (sec) | (sec) | (sec) | (sec) | (sec) | (sec) |
| Guidance Go-Inertial | -8 | -8 | -8 | -8 | -8 | -8 |
| RD-180 Ignition | -2.7 | -2.7 | -2.7 | -2.7 | -2.7 | -2.7 |
| SRB Ignition | -- | -- | 0.8 | 0.8 | 0.8 | -- |
| Liftoff | 1.10 | 1.10 | 1.05 | 1.04 | 1.04 | 0.97 |
| CCB Throttle Down/Up for max Q | -- | -- | 38/58 | 39/57 | 41/57 | 59/312 * |
| Strap-On SRB Jettison | -- | -- | 118 | 116 | 117 | -- |
| LRB Engine Cutoff | -- | -- | -- | -- | -- | 228 |
| LRBs Jettison | -- | -- | -- | -- | -- | 236 |
| PLF Jettison (500 series and HLV) | -- | -- | 212 | 206 | 210 | 298 |
| CCB Engine Cutoff (BECO) | 242 | 238 | 249 | 251 | 252 | 367 |
| CCB/Centaur Separation | 250 | 246 | 257 | 259 | 260 | 375 |
| Centaur Main Engine Start 1 (MES1) | 260 | 256 | 267 | 269 | 270 | 385 |
| PLF Jettison (400 series) | 268 | 264 | -- | -- | -- | -- |
| Centaur Main Engine Cutoff (MECO1) | 946 | 1,015 | 904 | 1,041 | 782 | 563 |
| Start Turn to MES2 Attitude | 1,103 | -- | 1,032 | 7,871 | 983 | 818 |
| MES2 | 1,473 | -- | 1,402 | 8,241 | 1,353 | 1,188 |
| MECO2 | 1,689 | -- | 1,668 | 8,373 | 1,617 | 1,678 |
| Start Turn to MES3 Attitude | -- | -- | -- | -- | 19,893 | 19,957 |
| MES3 | -- | -- | -- | -- | 20,263 | 20,327 |
| MECO3 | -- | -- | -- | -- | 20,391 | 20,543 |
| Start Alignment to Separation Attitude | 1,691 | 1,017 | 1,670 | 8,375 | 20,393 | 20,545 |
| Separate Spacecraft | 1,858 | 1,184 | 1,837 | 8,542 | 20,560 | 20,712 |
| Start Turn to CCAM Attitude | 1,883 | 1,209 | 1,862 | 8,567 | 20,585 | 20,737 |
| Centaur End of Mission | 6,258 | 5,584 | 6,237 | 12,942 | 24,960 | 25,112 |
| * HLV CCB throttling is for performance efficiency and not for max Q control | | | | | | |

This timeline will only be used until centaur main engine cut off. The Atlas V HLV will have three centaur main engines to bring the primary payload into GSO; the SPLs have to be deployed before the deployment of the PPL.

The important events in the timeline are the engine cut off of the two LRB (228 seconds) and their deployment at 236 seconds. This event is followed by the jettison of the payload fairings and the engine cut off of the common core booster (367 seconds). The separation of the CCB and the centaur stage starts at 375 seconds and the first centaur main engine starts 10 seconds later until it is cut off at 563 seconds.

### C. ARTICULATION FILE

Articulation keyword blocks and commands are an integral part of a model file. A separate file, called the model articulation file, contains the information needed to animate the articulation commands contained in the model file. The model articulation file contains a time-ordered list of data about each articulation that can be applied to the associated object [1].

Articulation file names have to be identical to the name of the object (not the model file), and have an object-specific extension. In this simulation, the object for which articulations are being specified is the launch vehicle called `CubeSat-LV` and the model file is `atlas-v-heavy-NPS-SCAT.mdl`, the articulation file name is `CubeSat-LV.sama` and is stored in the same folder like the scenario.

### 1. How to create an articulation file

An articulation file can be created with any common text editor. The only important thing is to save the file as explained above in the scenario directory.

The standard model format consists of the following commands:

```
NEW_ARTICULATION
STARTTIME 0
DURATION 300
DEADBANDDURATION 0.000000
ACCELDURATION 0.000000
DECELDURATION 0.000000
DUTYCYCLEDELTA 0.000000
PERIOD 0.0
ARTICULATION SolarPanel
TRANSFORMATION Roll
STARTVALUE 0
ENDVALUE 45
```

In this example a single articulation that starts at Epoch Time `0`, runs for `300` seconds, and moves a model component called `SolarPanel` from `0` degrees to `45` degrees is created.

An articulation begins its transformation at the STARTTIME. The transformation itself is defined by the STARTVALUE, which specifies the value of the transformation at the beginning of a cycle, and the ENDVALUE, which specifies the value of the transformation at the end of a cycle. The time required to complete one cycle is specified by the PERIOD. The articulation ends at ⌈STARTTIME+DURATION⌉. Depending on the values specified for PERIOD and DURATION, the articulation can cycle any number of times.

After defining the STARTVALUE and ENDVALUE, the manner in which the transformation value changes can be defined [1]:

- Linear: By default, the transformation value varies from the STARTVALUE to the ENDVALUE over a cycle

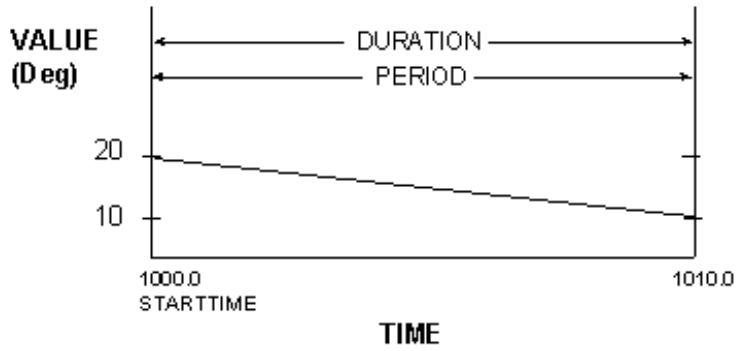



**Figure 21: Linear cycle**

- Step: To define a step cycle, the DUTYCYCLEDELTA has to be greater than 0.0. The DUTYCYCLEDELTA is an offset from the beginning of a cycle where the transition from STARTVALUE to ENDVALUE will occur.

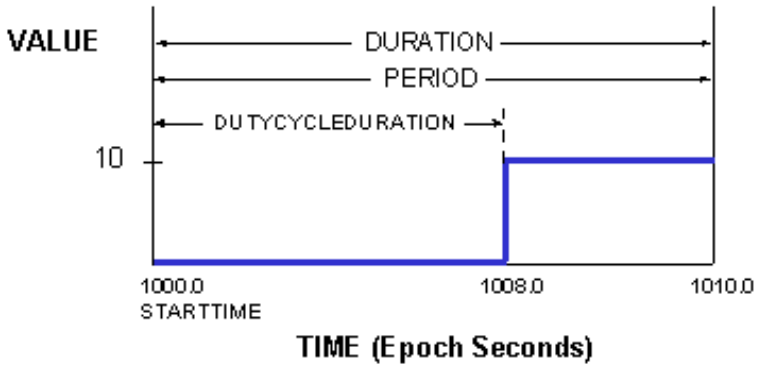



**Figure 22: Step cycle**

- Nonlinear: To define a nonlinear cycle, set the ACCELDURATION or DECELDURATION value to be greater than 0.0. The value of ACCELDURATION plus DECELDURATION must be less than or equal to the PERIOD. From the [Beginning of a PERIOD] to the [Beginning of a PERIOD + ACCELDURATION], the value changes under constant acceleration. From the [Beginning of a PERIOD + ACCELDURATION] to the [End of the PERIOD - DECELDURATION], the value changes under zero acceleration. From the [End of the PERIOD - DECELDURATION] to the [End of the PERIOD], the value changes under constant deceleration.

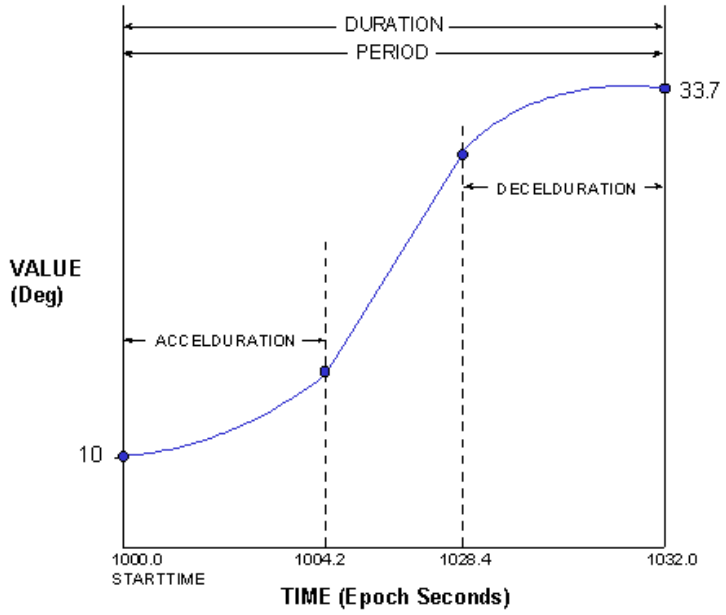



**Figure 23: Nonlinear cycle**

- Deadband: You can use a DEADBANDDURATION parameter to lessen the time it takes to move from STARTVALUE to ENDVALUE. If a DEADBANDDURATION parameter is specified in the articulation keyword, the ENDVALUE is reached at [End of a PERIOD - DEADBANDDURATION]. From the [End of a PERIOD - DEADBANDDURATION] to the [End of PERIOD], no movement occurs. The DEADBANDDURATION parameter is used in conjunction with linear, step and nonlinear cycles.

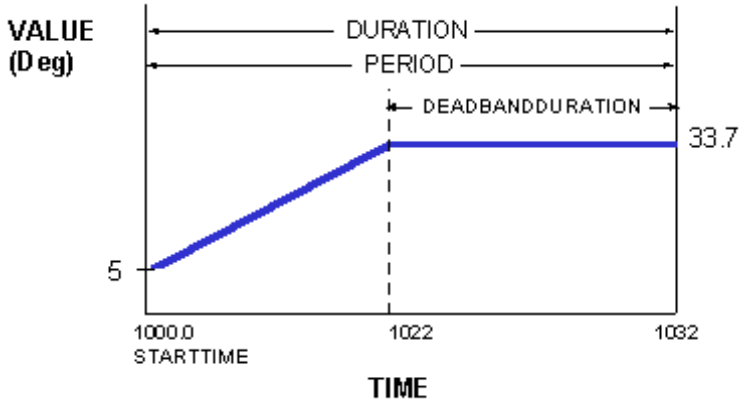



**Figure 24: Deadband cycle**

### 2. Articulation file CubeSat-LV

The articulation file `CubeSat-LV.sama` is the file that animates the articulation of the CubeSat-LV model (III.A.3) in STK.

Line 1 – 13: Description

Line 14 – 46: BOOSTER_AUX1_FLAME

Based on the timeline of the Atlas V launch vehicle the flames of the three boosters are switched on (0 →1). The duration of these sequences is only 0.01 seconds, because the boosters reach full power immediately. The flames of the LRBs are turned off at 228 seconds (see Table 1).

Line 27 – 79: BOOSTER_AUX2_FLAME

This is the definition of the animation for the second LRB, which is equal to the BOOSTER_AUX1_FLAME.

Line 80 – 112: BOOSTER_MAIN_FLAME

This articulation is for the flames of the main engine (CCB), which is cut off after 367 seconds. The cycle is linear, as defined before.

Line 113 – 167: FAIRING1

`Fairing1` is one of the two PLF on top of the Atlas V launch vehicle. The have to jettison before the separation of the centaur stage and the CCB. Therefore, the transformations `rotate`, `separate`, `drop`, and `size` are used. There was no presetting given for these transformations that is why the DURATION, DEADBANDDURATION, and ACCELDURATION are random values. They were adjusted by animating the whole simulation to assure a realistic movement.

Line 168 – 221: FAIRING2

`Fairing2` is the second PLF and at the opposite position of Fairing1. The transformations are basically the same, only the rotation and separation are in the other direction.

Line 222 – 275: Booster_Aux1

The jettison of the two LRBs starts at 236 seconds and the transformations are equal to the separation of the PLF. The size of the LRB is scaled down after the rotation, separation, and drop to create the effect of a removing action.

Line 276 – 330: Booster_Aux2

The articulations for this LRB are equal to `Booster_Aux1`, only in the opposite direction.

Line 331 – 359: SEPERATION FIRST STAGE AND SECOND STAGE

Eight seconds after the cut off of the CCB, the centaur stage separates from the CCB. This is realized by dropping the `Booster_Main` component and by simultaneous scaling.

Line 360 – 392: ENGINE CENTAUR STAGE MES1

The first main engine of the centaur stage starts 10 seconds after the separation and is cut off at 76 seconds. This sequence is simulated by scaling the `Centaur_Flame`.

Line 393 – 425: ENGINE CENTAUR STAGE MES2

A second burn of the centaur stage is necessary to reach a circular orbit. The burn time (20 seconds) is calculated in III.E.2.b).

Line 426 – 480: OPENING P-POD

In order that the CubeSat can come out of the P-POD, the removable door has to be opened. Usually all P-PODs are loaded and they will be opened at a certain time, but in this scenario only `Lid1, Lid3, Lid6,` and `Lid8` will open. The clearance between the opening sequences is 10 seconds. All these values are randomly picked to demonstrate one possible scenario.

Line 481 – 525: SEPERATION OF NPSCuL and NPS-SCAT

The mission of NPS-SCAT will start with a push by a spring right after the opening sequence of the P-POD. The detaching of NPSCuL and the other three CubeSats is animated by the drop transformation. To give the impression of movement, the size of the centaur stage is scaled down.

## D. SCENARIO

### 1. How to create a scenario

A scenario in STK is an instance of an analytical or operational task that you are modeling with STK. When you select `Save` from the `File` menu in STK, you are saving a scenario. However, unlike other applications that save a single file, STK saves a scenario as a group of files that comprise the collection of objects that are relevant to the scenario. When you save a scenario, the scenario itself is saved as an object, and each object within the scenario is saved individually. The file extension for the scenario object is `.sc`. Additionally, the current layout of the workspace is saved in a Workbook file.

Satellites, aircraft, ground vehicles, ships, planets, stars, receivers and transmitters are examples of the objects that can constitute a scenario in STK.

Within a scenario, you can set general properties such as time period, units of measure, and the appearance of the 2D and 3D windows. Information that is contained in the scenario can be used to create reports and to perform other analytical tasks. It is possible to create an unlimited number of scenarios with STK; however only one scenario can be open at a time [1].

### 2. NPS-SCAT scenarios

The whole lifecycle of the NPS-SCAT CubeSat will be simulated. Therefore, the launch and deployment sequences (NPS-SCAT_mission) is separated from the in-space operations (NPS-SCAT_com).

## E. SCENARIO NPS-SCAT_MISSION

Basically, the `NPS-SCAT_mission` scenario consists of the Space Launch Complex 41 at Cape Canaveral Air Force Station, wherefrom the Atlas V HLV will launch and the satellite `CubeSat-LV` based on the model created in the `atlas-v-heavy-NPS-SCAT.mdl` file.

### 1. Facility Cape Canaveral

The facility `Cape_Canaveral_41` was created by clicking on `Insert` → `New` → `Facility`. The position of the Launch Complex 41 is 28°35'N and 80°34'58"W and is operated by the US Air Force. The settings can be modified by using the `Properties Browser` (right click on facility).

To create a more realistic simulation, an imagery of the facility can be created. This can be done by clicking on the `Globe Manager` icon (only

possible in the 3D graphics window). The next step is a right click on the Earth folder in the hierarchy and choosing Add Terrain/Imagery. It is possible to import imagery from local files or from the Globeserver. In the latter case, Globserver Access has to be enabled (Scenario → 3d Graphics → Globeserver).

An inconsistency arose when the imagery for the Air Force base was included. The configured position of the launch pad was not a part of the map. Therefore, the longitude and latitude was modified until both settings agreed. Another 3D setting is the appearance of the facility. The model file of the Atlas V launch pad (atlas-lp-modified.mdl) created in III.A.4 can be used. The file can be added by inserting the model file name in 3D Graphics → Model. It is also necessary to scale the model to zero; otherwise, the relation to the Atlas V launch vehicle would be deformed.

### 2. Satellite CubeSat-LV

After creating a satellite by clicking on Insert → New → Satellite, the Astrogator has to be selected from the list of Propagators.

STK/Astrogator is a specialized analysis module for interactive orbit maneuver and spacecraft trajectory design. Astrogator acts as one of the propagators available for a satellite object. The Astrogator calculates the satellite's ephemeris by executing a Mission Control Sequence, or MCS, that you define according to the requirements of your mission. Astrogator provides the ability to model impulsive and finite maneuvers as well as high-fidelity orbit propagation. It also provides a differential corrector Targeter, which allows you to find the necessary values of control parameters - such as launch epoch or burn duration - to meet desired mission goals. Furthermore, Astrogator allows

you to define automatic sequences, which represent pre-defined sets of actions that can be performed whenever a specified event occurs - such as a maneuver that occurs at every periapsis. Astrogator also includes a catalog and editor called the Component Browser. The Component Browser lets you define and customize engine models, force models, propagators, central bodies, atmospheric models, and other elements of a space mission analysis scenario. In addition, the Component Browser contains a wide array of calculation objects. All of these elements can then be used or adapted in any STK/Astrogator scenario [1].

### *a) Basic settings*

All of the important settings for this scenario are done with the Astrogator except for the attitude and the 3D model. The settings for the attitude are presented in III.E.2.c).

To specify the model the `3D Graphics → Model` tab can be used. It is possible to display the same model for the whole simulation, therefore the option `Model File` has to be checked. Afterwards the `atlas-v-heavy-NPS-SCAT.mdl` file can be selected. Another important issue is the scaling of the model. Usually the models are scaled two or four times to get a better display in the simulation. In this case, the scale is set to zero ($10^0$) because otherwise the results of the solar cell tool would not be correct.

### *b) Astrogator*

When you select Astrogator as the propagator for a satellite object, the Orbit properties page transforms to a specialized layout in which you can define the spacecraft's MCS.

The left side of the window is occupied by the MCS Tree - which lists the segments that make up the MCS and depicts their relationships to each other - and the MCS Toolbar - which contains buttons that perform various MCS and individual segment operations. By default, the MCS contains `Initial State` and `Propagate` segments, which produce a low-Earth orbit. The right side of the window is occupied by the parameters of the segment that is currently selected in the MCS Tree.

Beneath the MCS Tree is the `Results...` button, which allows you to specify calculation objects to be reported and targeted for each segment. Clicking this button will open the User-Selected Results window, in which you can select calculation objects to include in the summary report for the currently selected segment, and to target when defining a Differential Corrector Target Sequence profile. In addition to using the button, you can open the User-Selected Results window by right clicking on a segment and selecting `Results...` from the drop-down menu.

The `Initial` and `Final` fields beneath the segment parameters area are apparent for every segment in the MCS and serve the same purpose for each; the initial field displays the scenario time and date at the beginning of the currently selected segment, while the final field displays the scenario time and date at the end of that segment. If a segment has not yet been run, these fields will be marked "Not Set" for that segment - since these values are not determined until the segment is run [1].

Every time some values or settings are changed in the `Astrogator`, the `Run Entire Mission Control Sequence` button has to be clicked.

(1) Launch

The Launch segment can be used to model a simple spacecraft launch from earth or another central body. The trajectory that is produced is a simple curve rising vertically from the launch pad that turns over smoothly to insert the launch vehicle into orbit with a zero flight path angle at the insertion point using the specified velocity.

To define a Launch segment, you must define parameters for the launch trajectory, the launch location, the burnout point, and optionally the burnout velocity. Specific launch location and burnout point parameters are framed in separate sections of the window, and the burnout velocity parameters can be accessed from the burnout section [1].

The selected Central Body of this Launch segment is the Earth and the Step Size (time interval between calculated ephemeris output points) is set to 0.1 seconds. The Pre-Launch Time is set to 15 seconds to get some more time for the movie.

The easiest way to create the location of the launch is by clicking on the Select Facility icon, which allows you to select Cape Canaveral from the facility list of your scenario. It is also necessary to raise the altitude, because the center of the model is the center of the CubeSat. The Epoch field specifies the date and time of the launch.

After defining the general and launch location parameters, the location of the burnout point has to be defined. In this simulation the burnout point definition Launch Az/Alt is chosen, which results in a defined burnout point in reference to distance downrange along an azimuth, measured from the surface of the Earth or other central body. According to the timeline given in Table 1 the Time of Flight is 367 seconds. The fact that the two LRBs are cut off before is not considered. The Azimuth (angle defining the launch direction, measured from North towards East) is set to 60 degrees, the

Downrange Distance (distance along a great arc from the launch location to the burnout sub-satellite point) is 600 km, and the Altitude is 150 km (Figure 5). The motion along the ellipse (Ascent Type) is set to Cubic Motion, which means that the motion is computed based on given positions and velocities.

The option Use Fixed Velocity is used for the Burnout Velocity, where the arc between the launch and insertion positions determines the inclination of the final state of the launch segment, and the horizontal flight path angle is set to zero. The default value is 7.29976 km/sec.

The definition of the Satellite Properties based on the centaur stage, because the maneuver of this stage will be the main event of this model. Therefore, the dry mass of the spacecraft has to be calculated and added to the configurations.

$$m_0 = m_{CS} + m_{fuel} + m_{PL} \tag{3.1}$$

$$m_0 = 2,316\ kg + 20,860\ kg + 13,000\ kg = 36,176\ kg \tag{3.2}$$

$$m_{dry} = m_0 - m_{fuel} = 15,316\ kg \tag{3.3}$$

(2) LV_Separation-Orbit1

The Propagate segment can be used to model the movement of the spacecraft along its current trajectory until meeting specified stopping conditions. The segment uses the defined propagator to propagate the state, adding each point to the ephemeris as it goes. After each step, the segment checks to see if any stopping conditions were met during the step; If so, it then finds the exact point, within tolerance, where the stopping condition is

satisfied. From that point, the segment either executes an autosequence or stops the propagation and passes the state at that point to the next segment [1].

This segment describes the time between the CCB engine cut off and the start of the centaur main engine (18 seconds). The used propagator is the `Earth HPOP Default v8-1` (Propagator used to create trajectory). Stopping conditions define the point at which the segment ceases propagation; in this simulation, the condition `stop after a specific duration` is used.

(3) DeltaV-12

This maneuver simulates the burn of the centaur stage. The `Maneuver` segment can be used to model a spacecraft maneuver. While the parameters for Impulsive and Finite Maneuvers are similar in many respects, there are important differences between them that warrant discussing them on individually.

The used `Finite Maneuver` is effectively a Propagate segment with thrust. It uses the defined propagator to propagate the state, accounting for the acceleration due to thrust. The magnitude of the thrust vector is specified by the maneuver's engine model; the direction of the thrust vector is specified by the maneuver's attitude control. Like propagate segments, each point calculated by the propagator is added to the ephemeris, and propagation continues until a stopping condition is met. Once a condition is met, Astrogator then finds the exact point, within tolerance, where the stopping condition is satisfied. From that point, the segment either executes an autosequence or stops the propagation and passes the state at that point to the next segment.

The parameters on the `Attitude` tab can be used to define the direction of the maneuver, while the `Engine` tab can be used to define the magnitude and the nature of the propulsion. The `Propagator` tab

can be used to define the duration of the maneuver and the method of calculating its ephemeris [1].

The contents of the Attitude page are defined by the Attitude Control setting. In the current case, the thrust vector is specified in Cartesian form with respect to the thrust axes (Thrust Vector). The attitude is updated during the burn; this forces the thrust vector to the specified direction at every instant throughout the burn. The thrust vector therefore rotates with the specified coordinate system and points in the (1 0 0) direction (velocity, normal, co-normal).

The only setting that has to be modified in the Engine tab is the Engine Model. Usually a predefined model can be chosen from the list; in this case, a specific engine model for the Atlas V centaur stage will be created in the Astrogator Browser (View → Astrogator Browser). A new component is created in Astrogator by duplicating or 'cloning' an existing component and then editing the clone's properties. Therefore, the model Constant Thrust and $I_{sp}$ has to be highlighted in the component browser, followed by clicking on the Duplicate button. After renaming and commenting the new model, the components can be modified. The gravitational acceleration constant at sea level on the Earth is set to 0.00980665 km/s$^2$. According to the attributes given in Figure 6, the $I_{sp}$ is set to 450.5 seconds and the Thrust to 99.2 kN. All settings of the created engine model file are saved in the scenario directory (STK 8 → Astrogator → Engine_Models).

The Trip Value has to be set in the Propagator tab. Therefore an Excel file was created to calculate the Δv and the burn times necessary to reach a circular orbit (Appendix C). In the following, the calculations for the burn time are presented [6].

Orbit 1 (after launch sequence):

$$r_{a1} = 6522.47\ km$$
$$r_{p1} = 6035.44\ km$$

$$sma_1 = \frac{r_{a1} + r_{p1}}{2} \tag{3.4}$$

$$v_{a1} = \sqrt{2 * \mu_E * \left( \frac{1}{r_{a1}} - \frac{1}{2 * sma_1} \right)} \tag{3.5}$$

Orbit 2:

$$r_{a2} = 6728.14\ km$$
$$r_{p2} = r_{a1} = 6522.47\ km$$

$$sma_2 = \frac{r_{a2} + r_{p2}}{2} \tag{3.6}$$

$$v_{p2} = \sqrt{2 * \mu_E * \left( \frac{1}{r_{a2}} - \frac{1}{2 * sma_2} \right)} \tag{3.7}$$

$$v_{a2} = \sqrt{2 * \mu_E * \left( \frac{1}{r_{a2}} - \frac{1}{2 * sma_2} \right)} \tag{3.8}$$

$$\Delta v_{12} = v_{p2} - v_{a1} \tag{3.9}$$

Δv equation:

$$\Delta v_{12} = g_0 * I_{sp} * \ln \left( \frac{m_1}{m_{f12}} \right) \tag{3.10}$$

$$m_{f12} = \frac{m_1}{e^{\frac{\Delta v_{12}}{I_{sp} * g_0}}} \tag{3.11}$$

Thrust equation:

$$T = I_{sp} * g_0 * \frac{dm}{dt} \tag{3.12}$$

$$dt_{12} = \frac{m_1 - m_{f12}}{T * g_0 * I_{sp}} \tag{3.13}$$

The result of these equations is a burn time of 76.02 seconds and a Δv of 0.2135 km/s.

(4) Orbit 2

This propagator is used as a parking orbit until the launch vehicle reaches apoapsis (stop at the point farthest from the origin).

(5) DeltaV -23

A second burn is necessary to reach circular orbit of 350 km altitude.

Orbit 3:

$$r_{a3} = r_{a2} = 6728.14\ km$$
$$r_{p3} = r_{a2} = 6728.14\ km$$

$$sma_3 = \frac{r_{a3} + r_{p3}}{2} \tag{3.14}$$

$$v_{circ3} = \sqrt{2 * \mu_E * \left( \frac{1}{r_{a3}} - \frac{1}{2 * sma_3} \right)} \tag{3.15}$$

$$\Delta v_{23} = v_{circ3} - v_{a2} \tag{3.16}$$

Δv equation:

$$\Delta v_{23} = g_0 * I_{sp} * \ln\left( \frac{m_{f12}}{m_{f23}} \right) \tag{3.17}$$

$$m_{f23} = \frac{m_{12}}{e^{\frac{\Delta v_{23}}{I_{sp} * g_0}}} \tag{3.18}$$

Thrust equation:

$$T = I_{sp} * g_0 * \frac{dm}{dt} \tag{3.19}$$

$$dt_{23} = \frac{m_{f12} - m_{f23}}{T * g_0 * I_{sp}} \tag{3.20}$$

The burn time of the second maneuver is 20.7 seconds and the Δv is 0.0599 km/s.

(6) Orbit 3

After reaching the circular orbit, a time delay of 300 seconds is included. This value is randomly chosen and can be modified depending on the further mission of the centaur stage.

(7) CubeSat Separation

This maneuver describes the deployment of NPS-SCAT. Therefore, the impulsive maneuver type is chosen, because the CubeSat will only be accelerated once by the spring inside of the P-POD. The Δv is set to 0.002 km/s.

(8) Orbit 4

This last propagator is necessary to keep the model in the simulation for one day.

***c) Attitude***

After the deployment, the CubeSat probably will rotate or tumble. In this case, the model should be fixed, because otherwise the launch vehicle would start tumbling right after the launch. The attitude type (`Properties Browser → Basic → Attitude`) is set to `ECF Velocity Alignment with Radial Constraint`. The vehicle's

X-axis is aligned with the Earth-fixed velocity vector direction and the Z-axis is constrained in the radial direction (the direction along the position vector and opposite to geocentric nadir). The Constraint Offset angle can be used to introduce an additional rotational offset about the Earth-fixed velocity vector. This angle is measured in a right-handed sense (e.g. to constrain with the Y-axis set the offset to +90 degrees). In this case, the offset is set to zero degrees.

## F. SCENARIO NPS-SCAT_COM

It would have been too complicated to include all of the in-space operations to the NPS-Mission scenario. That is why a separate scenario (NPS-SCAT_Com) was created, which includes the power budget and the lifetime calculation of the CubeSat.

### 1. Facility Monterey (NPS)

The facility in Monterey is the location of the Naval Postgraduate School and the Space Systems Academic Group. That is why the communication with the satellite will take place from this location.

#### *a) Facility*

The facility of the NPS can be added the same way like the Cape Canaveral facility. This time the database for cities can be used by clicking on Insert → City from Database. The box City Name has to be checked and then "Monterey" can be typed in. There is only one result for this search and it will be inserted by hitting the OK button.

The appearance of this facility can also be changed with the help of the Properties Browser. This time the attributes in the 2D Graphics are modified, because the main task of this facility will be the transmitting of data. Therefore, the Marker Style under the Attributes is changed to the sensor symbol. This also has to be done for the 3D Graphics by using the 2.5 times scaled ground-antenna.mdl file.

### *b) NPS-Dish*

To include the dish of the SSAG, which is on top of the building Spanagal Hall, a sensor and a receiver has to be created and assembled to the Monterey facility. Therefore, the ground station has to be highlighted in Object Browser and then the sensor can be added by clicking on Insert → New → Sensor.

The settings of the sensor can be modified in the Properties Browser. In this simulation a half power sensor is used, which assures that the satellite has still a half power spot in the worst case. The frequency of the sensor is 2.44 GHz and the diameter of the dish is 10 ft (3.048 m). This results in a half angle of 1.412 degree (Basic → Definition). The pointing of the sensor can also be arranged and therefore the Targeted pointing type is selected. The following settings are used for this simulation.

Boresight Type: Tracking (aim a sensor at one or more targets)

Track Mode: Transpond (antenna points to the true location of the target object)

About Boresight: Rotate (indicates rotation about the sensor's or antenna's Z axis by the azimuth angle, followed by rotation about the new Y axis by 90 degrees minus the elevation angle)

The sensor should always track the NPS-SCAT CubeSat. That is why this satellite has to be assigned from the available targets.

### *c) Dish-Receiver*

The object `Dish-receiver` describes the receiver of the 10 ft dish of the SSAG. Therefore, the Simple Receiver Model with the following settings is used:

g/T: Receiver gain divided by the system noise

Frequency: Enter the desired value or select Auto Track

Bandwidth: Enter the desired value or select Auto Scale

Polarization: Describing the orientation of the electric field vector with reference to the antenna's orientation

Rain Outage: Estimated amount of degradation (or fading) of the signal when passing through rain

Link Margin: Specify a desired Link Margin Type from a predetermined set of types (BER, RIP, C/N, etc.) as well as a Link Margin threshold Value

Additional Gains and Losses

Spectrum Filter

Right now it is not possible to set the actual values for the receiver. That is why the default values will not be modified.

### 2. Satellite NPS-SCAT

The model file `NPS-SCAT.mdl` is used in the 3D graphics settings. This model also has to be scaled down to zero so that the results of the solar panel tool are useful.

#### *a) Basic Settings*

The NPS-SCAT CubeSat was inserted as described in the prior chapters. The basic settings for this spacecraft can be taken from the results of the ΔV calculations (Appendix C). The semi-major axis is 6728.14 km, the eccentricity is zero degree, and the inclination is 40 degree. This time, the satellite will tumble; this can be defined in the `Attitude` tab. Right now, it is unknown about which axis and with which spin rate the CubeSat will fly in-space. The chosen values are randomly picked and will be modified for the power budget analysis.

### *b) Antenna*

As shown in Figure 9 NPS-SCAT will have an antenna attached to the side panel. Right now it is not sure, what kind of antenna will be used to transmit the collected data to the ground station. That is why a usual deployable antenna was picked for the ongoing simulation.

The antenna (`Insert → New → Sensor`) is a simple conic one with a cone angle of 70 degree. This value is random, based on other common satellite antennas. The position of the cone is the Center of the satellite, which means that the cone rotates with the CubeSat.

### *c) Transmitter*

It is necessary to add a transmitter for sending data to the ground station, because the antenna sensor is just creating the cone. Therefore, a Simple Source transmitter is added to the sensor (`Insert → New → Sensor`). The actual data for the NPS-SCAT transmitter are still under development, it is only known that the frequency will be 2.44 GHz.

## 3. NPS-SCAT access times

STK allows you to determine the times one object can "access," or see, another object. In addition, you can impose constraints on accesses between objects to define what constitutes a valid access. These constraints are defined as properties of the objects between which accesses are being calculated. STK can calculate access from all types of vehicles, facilities, targets, area targets and sensors to all objects (including planets and stars) within a scenario.

The two objects for which the access is computed define an access. Once an access is created, it maintains a close relationship with the defining objects. If either one of the defining objects is changed in such a way that the access times may be altered, the access is automatically recomputed. In addition, if either one of the defining objects is removed from the scenario, the access is automatically removed.

To open the Access tool highlight the facility Monterey from which access will be calculated in the Object Browser, and select Access from the object menu on the Menu bar. When the Access tool opens, select the NPS-SCAT CubeSat for access computations in the Associated Objects list, and then click Compute. An asterisk (*) appears to the left of the object(s) being accessed to indicate that access calculations have been performed. If the Static Highlight option is on, the graphics window updates to display access from the first object to the second object based on time and object constraints defined. Access between the two objects appears as thick lines overlaying the ground tracks [1].

As it is shown in Figure 25, the ground station of the NPS has between six and seven access times for transmitting data. These times are only useful for the first considerations, because it is assumed that the satellite has an isotropic antenna and so the orientation of the antenna does not influence the access.

The access times can also be saved as a report, which will be necessary for the power budget calculations. It should be emphasized that these times are only an evidence for the visibility of the spacecraft. This does not mean that the link can be closed.

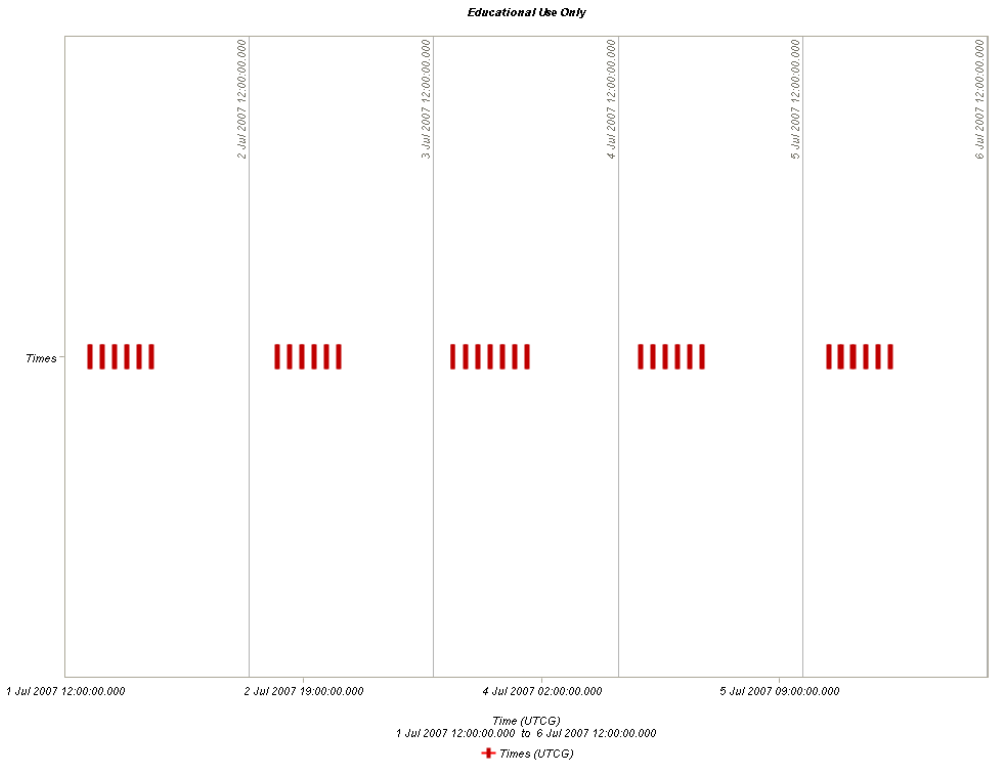



**Figure 25: Access times between ground station and NPS-SCAT**

### 4. NPS-SCAT solar power

The Solar Panel tool enables you to model the exposure of spacecraft-mounted solar panels over a given time interval, such as an orbit period. The result of the analysis can be used to determine varying availability of electrical power for operations to be performed by the spacecraft and onboard apparatus.

To compute the electrical power captured by the solar panels at a given point in time, the Solar Panel tool applies the following Basic Power Equation:

$$Power = Efficiency * Soalr\ Intensity * Effective\ Area * 1358 \frac{W}{m^2}. \qquad (3.21)$$

The Efficiency is specified for the solar panel in the satellite model file and ranges between 0 and 1 (III.A.2), Solar Intensity ranges from umbra (0) through penumbra (0 < i < 1) to full sunlight (1), and Effective Area is the cross-sectional area defined above.

The Solar Panel tool computes solar illumination over time by animating the scenario and periodically counting the pixels corresponding to illuminated portions of the solar panels under consideration [1].

To access the Solar Panel tool for a particular satellite, select one of the following highlight the satellite in the Object Browser and select `Solar Panel` from the Satellite menu on the Menu bar. In the new opening window, the Solar Panel Group `npsscat` has to be highlighted and the `Bound Radius` has to be set to 1 m. The data for the start and stop time and the time step can be adjusted for the events under consideration. The results can be generated as a report or as a graph.

Figure 26 shows the power that was generated by the solar cells of NPS-SCAT for one orbit. In this case, the CubeSat is tumbling and that is why the power is not a constant value. The four upper peaks represent the four different side panels while the lower peaks are representing the case when the sun is shining on one edge and three solar panels are illuminated, but the angle of illumination results in inefficiently.

The results can be stored as a report and can be exported. This is necessary for the power budget calculations in III.F.5. Therefore, after generating the report, it can be saved under `File` → `Export` → `Report`. In this case, the reports are saved as `.csv` files because these formats will be supported by MATLAB.

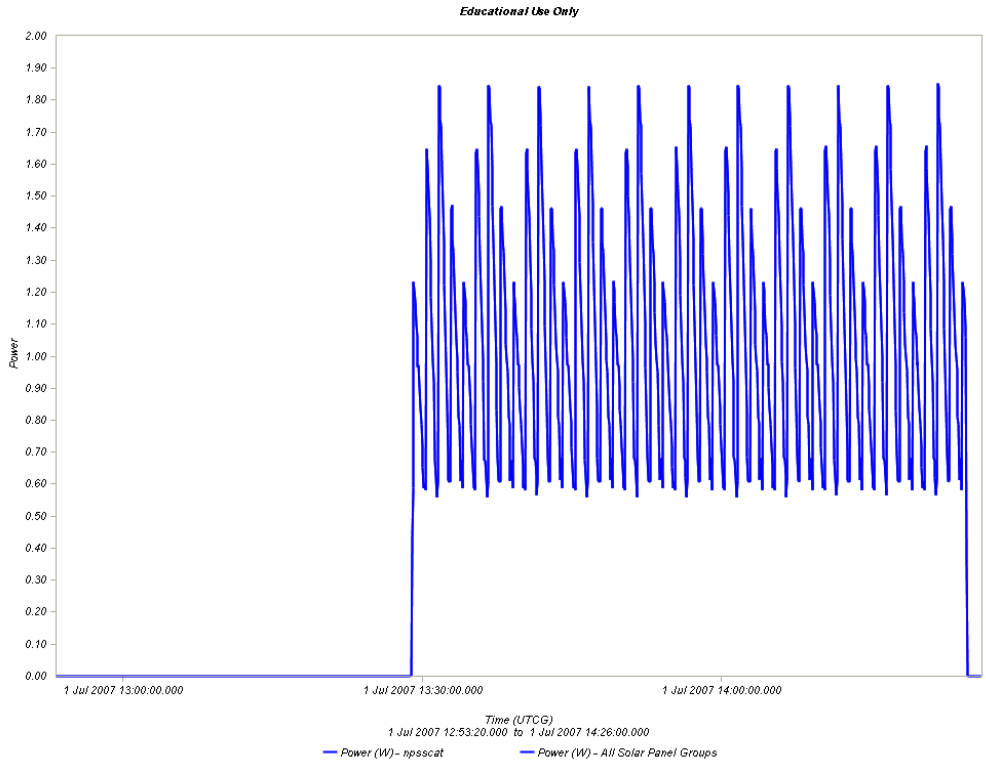



**Figure 26: Power created by the NPS-SCAT solar cells**

The fact that the NPS-SCAT CubeSat does not have six equal solar panels and that the spacecraft can spin about almost every possible axis, the average power created by the solar cells varies. Figure 27 shows the different results and the difference between the worst (0 0 1) and the best (1 0 0) case is 0.329 W.

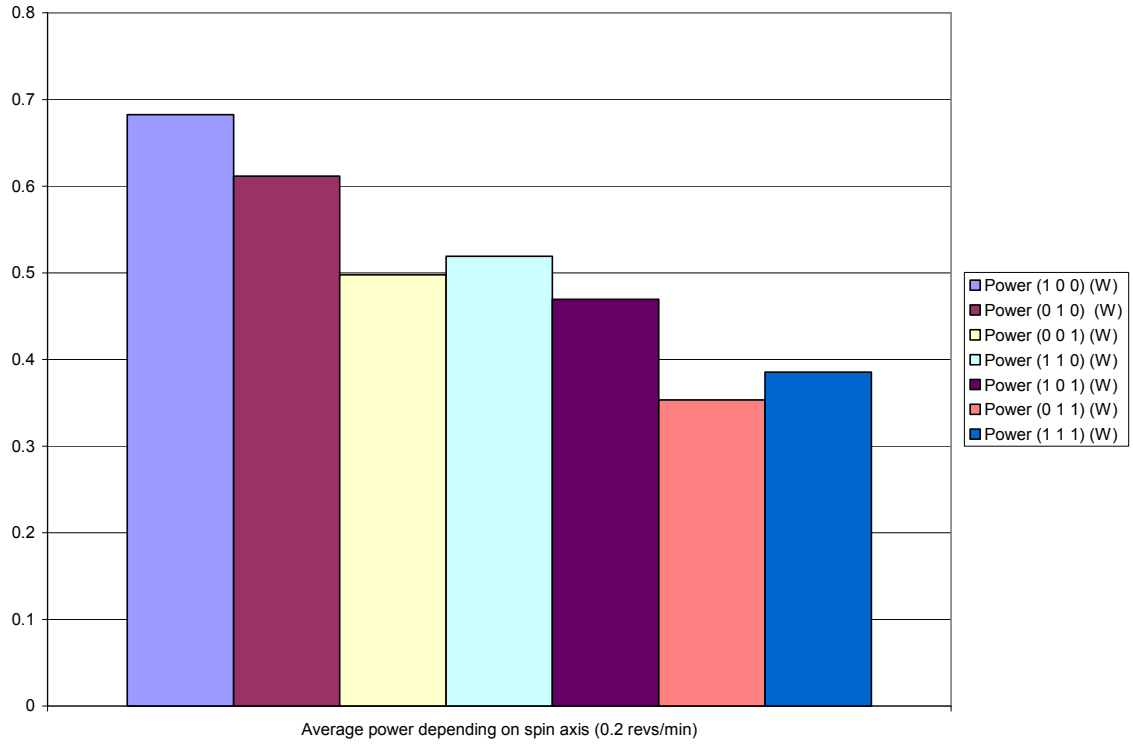



**Figure 27: Average power of the solar cells depending on the spin axis**

## 5. NPS-SCAT power budget

With the help of the access times and the report of the power generated by the solar cells, a power budget for the NPS-SCAT CubeSat can be created. This will show if the solar cells can create enough power to charge the batteries or if the communication board is taking too much power from the batteries. Basically the incoming power has to be higher than the outgoing power (positive trend of the battery state of charge).

A MATLAB file `power_budget.m` was created. The solar power report has to be saved as `power_npsscat.csv` and the satellite access times as `link-budget.csv`. Thereby it is very important to save change the `DateFormat` has to be set to `EpochMinutes` (`Report → Units`). Additionally, the link-budget file has to be modified. The start and

stop time of the accesses should have no decimal places, otherwise the MATLAB program cannot read the data.

After importing the data from the STK reports, a simple step-function is created with the data from the access times. Initially the link vector is filled with zeros, which represents no access. Afterwards a `for` function is created to insert ones for the duration of the access times (line 26 – 34).

Between the lines 36 and 54 the loads of the NPS-SCAT CubeSat are defined. The three loads are the main processor board (`mpb`), the communication board (`com`), and the solar cell measurement system board (`sms`). The different values for the sleep and the ON mode are given in Table 2.

**Table 2: NPS-SCAT loads**

| | Sleep mode [W] | ON [W] |
|---|---|---|
| Main processor board | 0.01 | 0.1 |
| SMS board | 0 | 0.15 |
| Communication board | 0.01 | 2.0 |

The link vector is the critical one for the main processor and the communication board. If this vector is unequal to zero (existing connection to ground station), the loads are rising from 0.01 W to 0.1 W respectively from 0.01 W to 2.0 W. The state of the SMS board depends only on the solar intensity, which is saved in the third column of the power file. If this value is one it means that the solar cells are generating power and the SMS board is taking data.

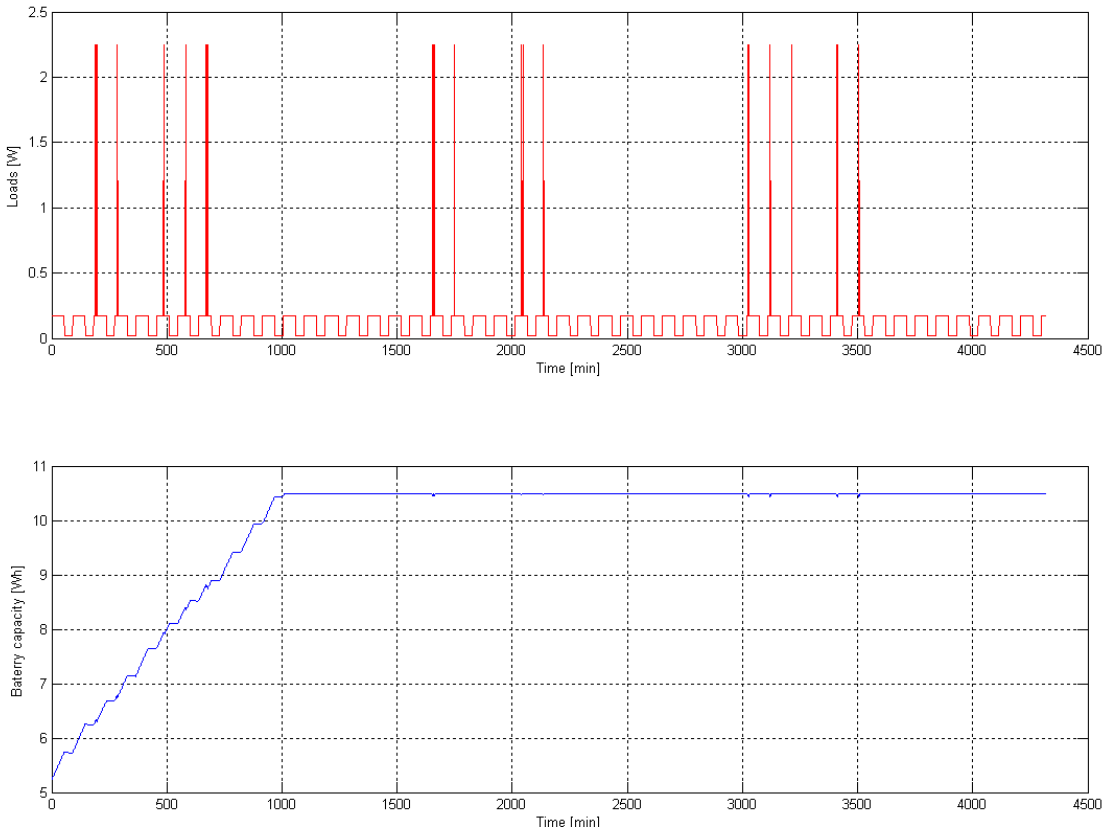



**Figure 28: NPS-SCAT power budget for 3 days**

The upper chart of Figure 28 shows the combined loads. The peaks are representing the communication board and the SMS board causes the step function. The graph is never zero, because even when the boards are in the sleep mode, they still need some power.

The power generated by the solar cells is converted to the `excess_power` vector. For our reference mission, the capacity of the on-board battery is 10.5 Wh and the efficiency of the conversion is estimated at 80%. The battery starts with a state of charge (SOC) of 50% and afterwards the actual SOC is calculated by adding the loads and excess energy times the efficiency. The lower graph of Figure 28 shows the battery capacity. The SOC reaches 100% after 1105 minutes and the power required by the communication board cannot create a negative trend.

Figure 29 shows a close-up of the power budget. The upper chart shows the loads and the lower one shows how the access times influence the characteristics of the battery SOC.

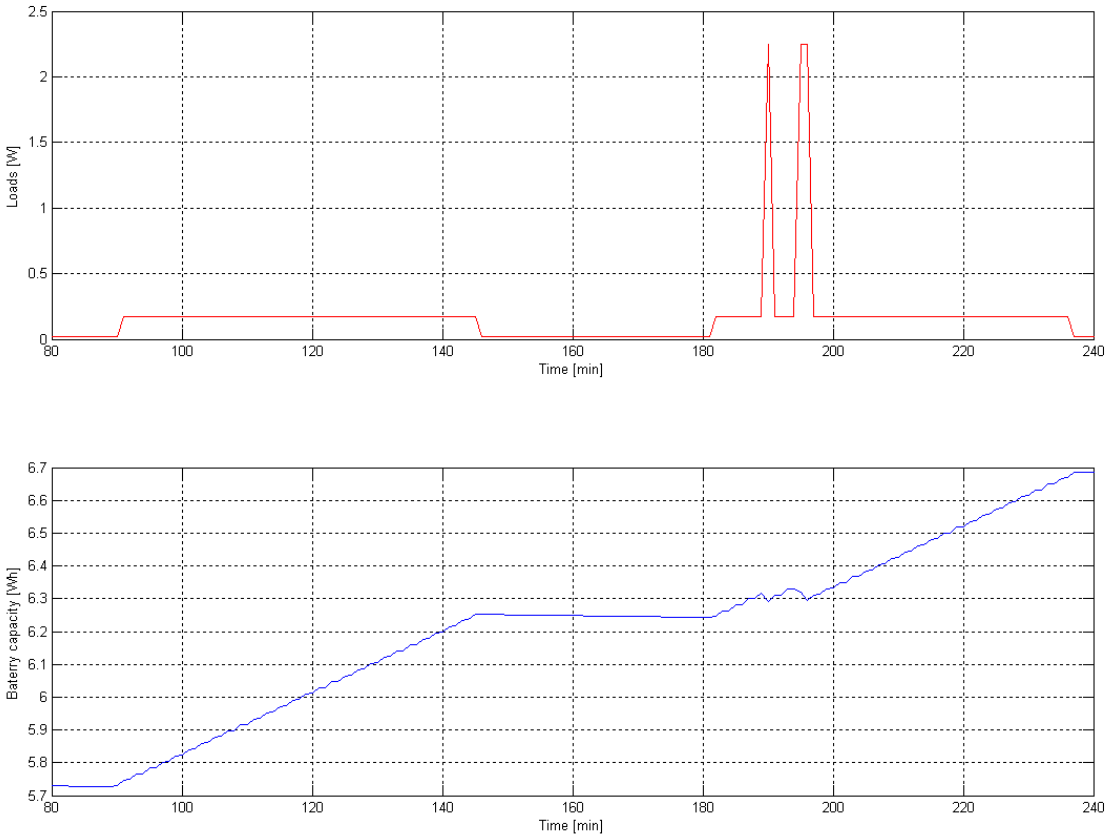



**Figure 29: Close-up of the NPS-SCAT power budget**

As mentioned in III.F.4, the power generated by the solar cells depends on the spin axis. Therefore, the MATLAB file `power_budget_compare.m` is calculating the battery SOC for the best (green), middle (red), and worst (blue) case. Figure 30 shows each graph in one chart and it becomes clear, that despite the worst case the SOC has a positive trend. It takes more time to charge the battery to 100%, but it is clear that the spin axis can be disregarded.

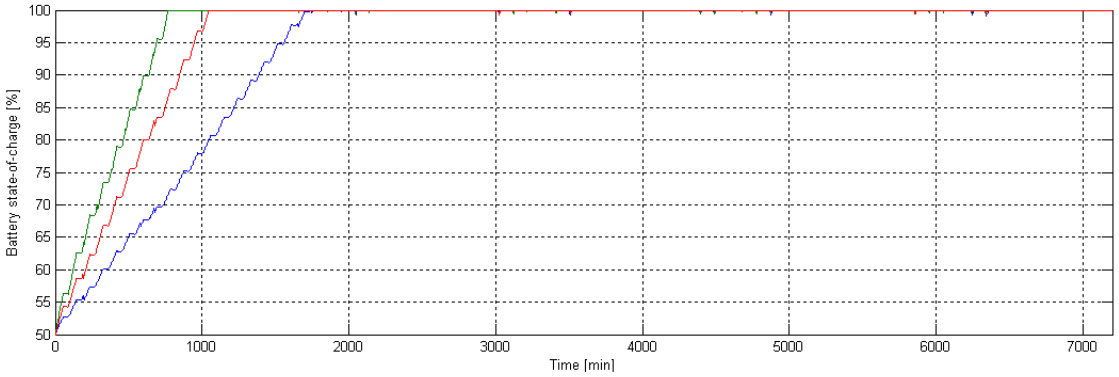



**Figure 30: Comparison of the battery SOC for different spin axis**

## 6. Deorbiting

Since Sputnik 1 launched on October 4th 1957, more than 28,000 objects were created to be launched into space. Right now over 9,000 of them are still in space and only 6% are functional spacecrafts. The remaining objects are so called space debris, which includes all space objects that are non-functional and man-made [7]. Figure 31 shows the Earth with the dots representing each piece of observable Artificial Debris.

Should the expected orbit of NPS-SCAT be high enough that it takes more than twenty-five years to deorbit, then it essentially becomes orbital debris after the end of the design mission life. In this case, a solution has to be found to deorbit the CubeSat. Therefore a small parachute, balloon or a long thin line are under development by different companies to solve this problem for CubeSats.

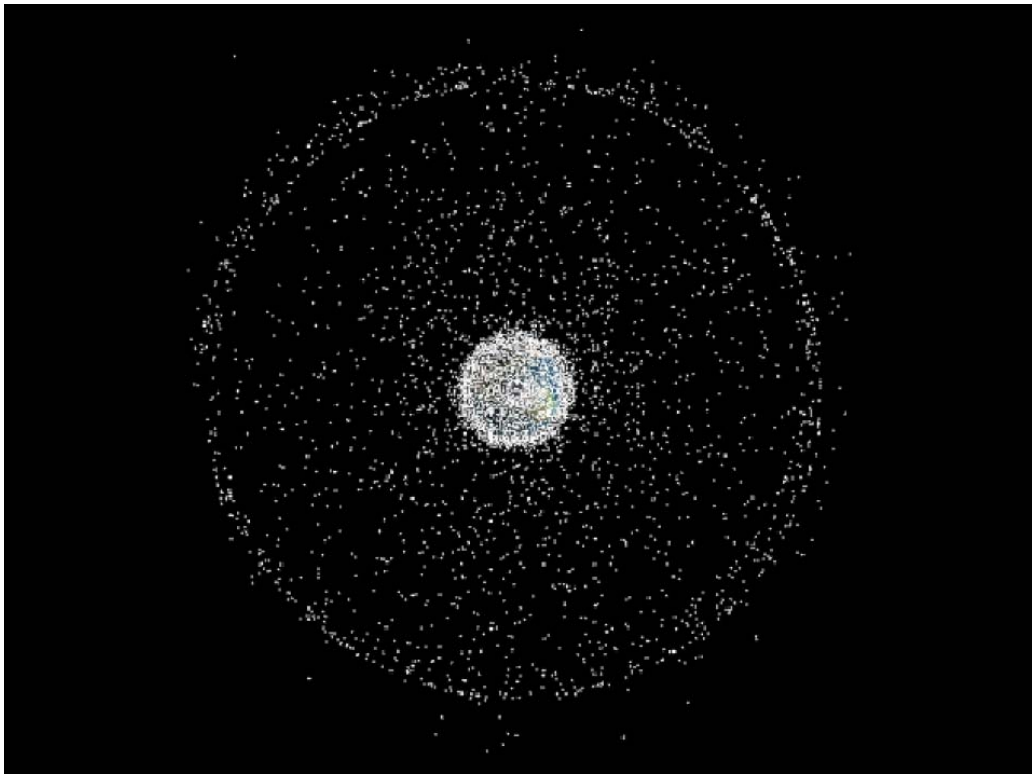


**Figure 31: Orbital debris around the earth [7]**

For our reference design mission, the effect of an added object to slow down the spacecraft is not necessary. In this simulation, the NPS-SCAT CubeSat has only an altitude of 350 km and the weight is one kg. That is why the satellite will deorbit by itself. With the help of the STK Lifetime tool, the estimated lifetime of the spacecraft can be calculated. The tool can be found under `right click on satellite → Satellite Tools → Lifetime`. The Lifetime tool estimates the amount of time a satellite can be expected to remain in orbit before atmospheric drag and other perturbations cause it to decay.

The most significant factors influencing the lifetime are the `Drag Coefficient`, the `Drag Area` and the `Mass`. The drag coefficient is not a constant value because of the tumbling motion. That is why the default value of 2.2 is used, which represents the average attributes of the satellite. The drag

area is a function of the shape of the CubeSat and is set to 0.01 $m^2$. This value is a reasonable estimate because the satellite is spinning.

After setting the `Satellite Characteristics`, a model for the `Atmospheric Density` has to be chosen. In this case, the NRLMSISE 2000 model is taken. This is an empirical density model developed by the US Naval Research Laboratory based on satellite data (2000 version, valid range of 0-1000 km). It finds the total density by accounting for the contribution of $N_2$, O, $O_2$, He, Ar and H, including anomalous oxygen above 500 km [1].

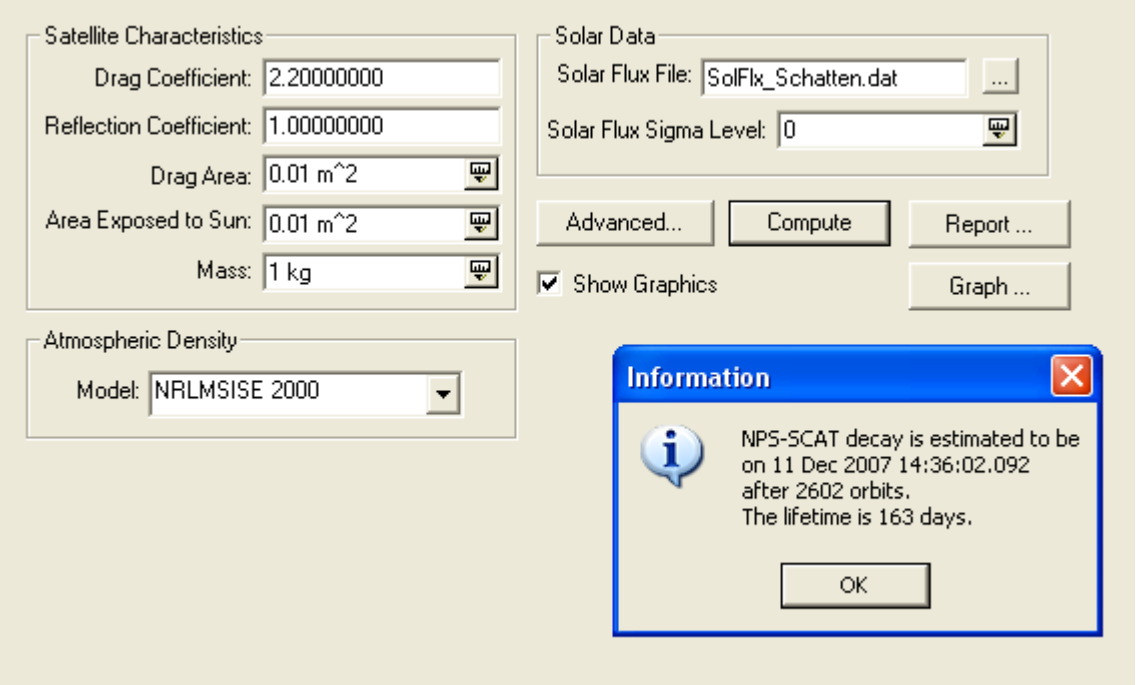



**Figure 32: STK Lifetime tool**

In the `Advanced` settings, the speed and accuracy of the calculations to be performed when estimating a satellite's orbital lifetime can be defined. To increase the performance of the Lifetime tool, number of `Orbits per Calculation` can be modified. The fewer orbits per calculation, the more precise the lifetime estimate is, but at the expense of compute time. The higher

the number of orbits per calculation, the less precise the lifetime estimate will be, but calculations are completed much faster. In general, this parameter is set to 10 for a quick estimate and 1 for the greatest accuracy [1]. Another important value is the Decay Altitude, that defines the altitude at which the satellite's orbit is determined to be decayed (lifetime calculations stop at this altitude). This value is set to 65 km. The results of the calculation can be observed after hitting the `Compute` button; the graph can be viewed after hitting the `Graph` button.

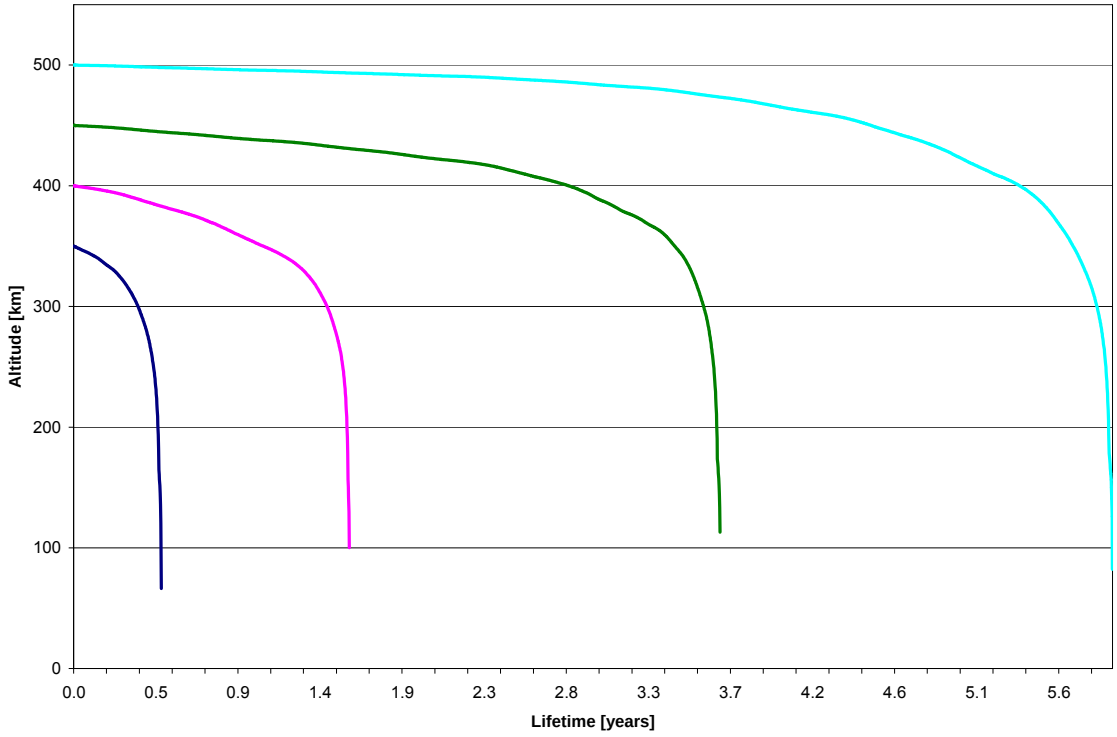



**Figure 33: 1U CubeSat deorbit from different altitudes**

Figure 33 shows the lifetime of a CubeSat depending on the initial altitude. The lifetime of the NPS-SCAT CubeSat in this simulation will be 163 days, starting at an altitude of 350 km.

It should be emphasized that although the Lifetime computations are based on sophisticated orbital theory and accurate environment models, the

result is still an estimate. Due to the seemingly random 10% variation in atmospheric density and because of the difficulty in accurately predicting solar activity, satellite lifetimes cannot be determined with accuracy better than +10% of the actual lifetime. Furthermore, assumptions and simplifications made in order to produce a practical computer implementation of the lifetime theory introduce an additional degree of uncertainty in the final result [1].

## G. CREATE A MOVIE

All of the animations are the results of the settings and calculations in STK. They can be used to create a movie of the whole lifecycle of the NPS-SCAT CubeSat to make the whole project clearer and easier to understand.

### 1. How to create movies in STK

There are a few important decisions to be made before you begin making an animated movie [1]:

1. Define the purpose
2. Define the audience
3. Select the medium
4. Create a storyboard

The first step in making a technical animation is to determine the reasons why you are making a technical animation. It has to be clear what things are important and who will watch the animations. Knowing the audience will help determine whether to make a highly conceptual movie (for managers) or a technically detailed movie (for engineers). The selected medium should most

effectively convey the message and the length of the movie will determine if it is for online presentations (under two minutes) or a medium with no time restriction. The final step should be the development of storyboard that shows all of the main events and a certain order.

After gathering information about the animations, the scenario should be technically accurate and esthetically pleasing. For example, the time step, articulations or the terrain for the 3D graphics can be adjusted.

The next step is to create a camera path, which defines the path of motion that the virtual camera takes during an animation. Click on the `View Path Editor` button to create a camera path and add a keyframe to it. Key-frames are points that the virtual camera flies through during animation. The camera path is a sequence of these key-frames that STK connects according to the method selected. Each keyframe stores the viewer position and orientation at a given time. The key-frames should be at points where the camera shows the very best images, such as seeing glints from solar panels or showing a feature of the earth in the background.

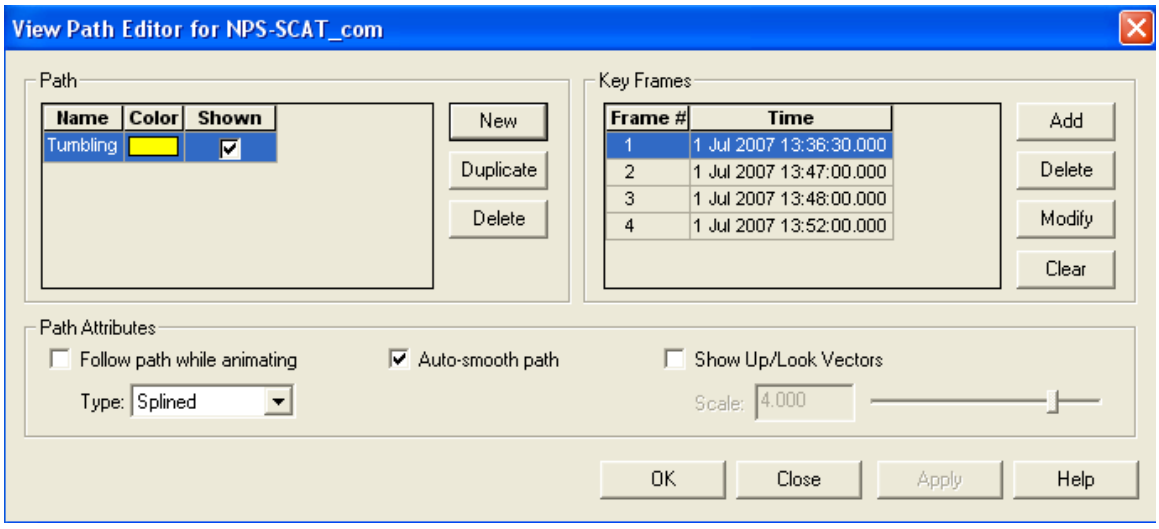



**Figure 34: STK Path Editor**

The Follow Path While Animating option allows you to evaluate the camera angles and transitions. When this option is selected, you cannot use the mouse to change the current view in the 3D Graphics window. It is recommended running the scenario to check if all keyframe transitions are satisfying, otherwise there might be unsmooth motions.

Another setting is the type of the path. A Splined path causes the animation to follow the view path spline for all viewer positions and viewer direction positions of the stored key-frames. In the Tethered option, the camera is attached to object and follows the view path. The object is always at the center of the frame, to change the distance between the camera and the object, move the viewer position closer to the object. The Auto-smooth path option corrects for anomalies in the object motion by optimizing the smoothness of the path between the keyframes. This option should always be selected.

After preparing the whole scenario for recording the movie, the Soft VTR properties page has to be opened (3D Graphics → Properties → Soft VTR) for the following steps [1]:

1. Enable Record every Animation Time Step.
2. Enter the name of the directory where you want to save your video.
3. Enter the name of your video in the Filename Prefix field.
4. In the File Format field, select the format appropriate for your movie. If you select the WMV format (recommended), you must also select a codec, framerate, and bitrate.
5. Enable Anti-Aliasing if this is your final recording. It is a good idea to not use Anti-Aliasing when recording your video for the first time. It causes the recording process to take longer and you may want to go back and make changes and then re-record.

6. Make sure that nothing is in front of the 3D Graphics window, including the Soft VTR page and the cursor. Anything in front of the 3D Graphics window will be recorded in your movie.
7. Click `Apply`.
8. Set the size of the window under the `Window Properties`, which depends on the purpose of the movie.
9. Click `Apply` and start animating forward. While it is running, the movie is being recorded. When the scenario gets to the end, it will stop. When it has stopped, go to the Soft VTR page, click `Off`, and then `OK`. The video is not properly recorded until you turn off the recording process and OK the Soft VTR page. If you try to open the file before this is completed, you will not be able to view your video.

The final steps in making a technical animation are testing in the user environment and editing the movie. What you see on the development PC is not necessarily what the user sees on his equipment. Test the movie in the user environment to ensure that the quality of the movie is maintained. The playback machine needs to contain the same codec that you used to record the video. When you are done recording, you can use video editing software to add music, tilting and annotation, cross-fades and scene dissolves or maybe still frames for special effects.

### 2. Creating and Editing the NPS-SCAT Movie

The animations for the NPS-SCAT lifecycle are split into the four main events: launch, separation of the launch vehicle, deployment of the CubeSat, and on-orbit operations. A particular camera path was created for each event and

afterwards recorded as separate movie files. Figure 35 to Figure 39 are showing the storyboard of the final NPS-SCAT movie.


**Figure 35: Atlas V HLV launch from Cape Canaveral AFS**


**Figure 36: Centaur stage separation at 150 km altitude**

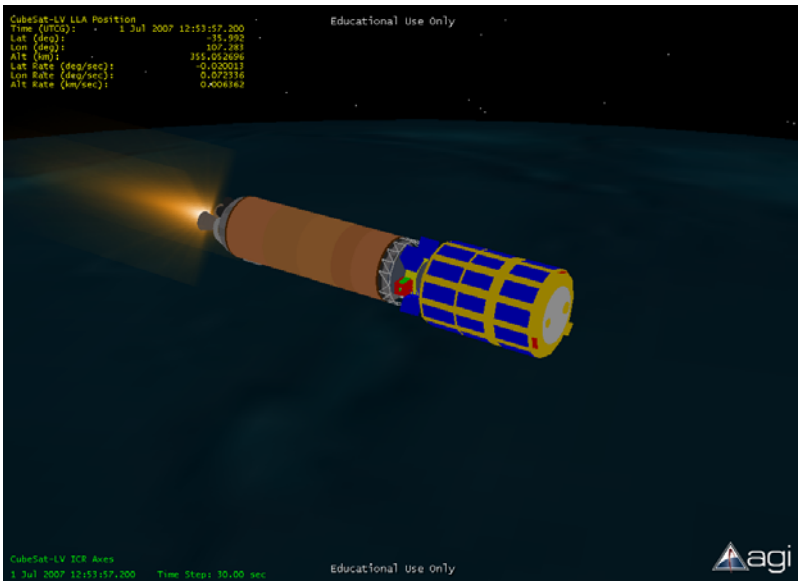



**Figure 37: Two burn maneuvers to reach a circular orbit of 350 km**

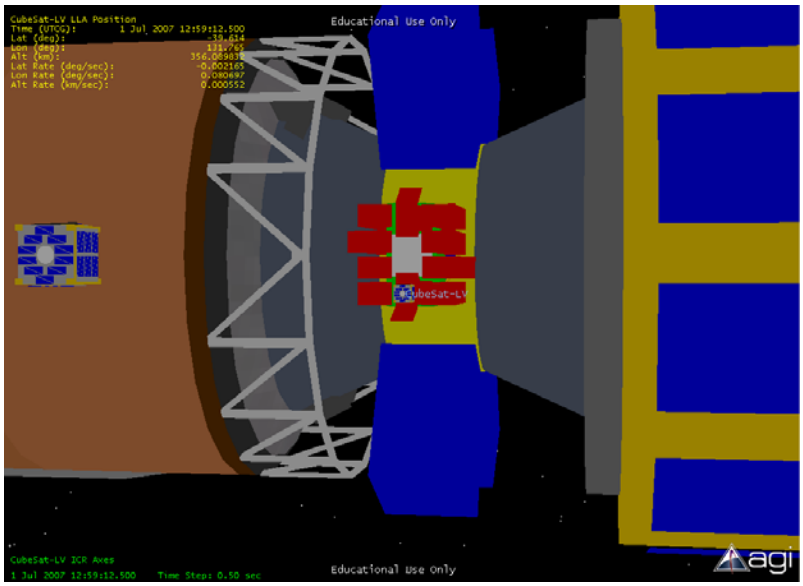



**Figure 38: CubeSat deployment from the NPSCuL**

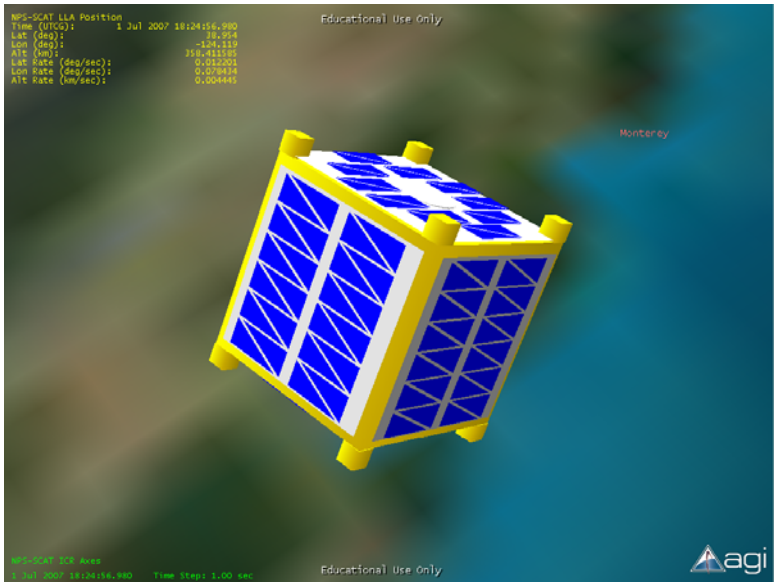



**Figure 39: NPS-SCAT on-orbit operations**

After creating the separate movie files, the editing software iMovie was used to create the NPS-SCAT movie. This software is a freeware and very easy to use, because the movies, pictures, and effects can be added to a project by drag and drop.

The movie starts with the logo of the Naval Postgraduate School and of the Space Systems Academic Group. Afterwards the main ideas of the two current CubeSat projects (NPSCuL and NPS-SCAT) are presented, followed by the motivation for this movie. This introduction is followed by the movies of the events as shown on the storyboard above. Finally, the movie concludes with the credits.

The final project was exported as a `wmv`-file for Windows user and as a `mov`-file for Mac user. The project was also exported as a DVD.

# IV. CONCLUSION

The results of the simulation of the lifecycle of the NPS-SCAT CubeSat are assembled in the NPS CubeSat Movie. This movie can be used to promote the basic ideas of the NPS CubeSat projects.

Another valuable result of this technical report is the production of the communication profile and the power budget for this satellite. Of course, certain, reference mission assumptions are used to create this first model, but the results are useful for the design and construction of the CubeSat. It is expected that when actual mission parameters such as orbit altitude and inclination are known and more accurate power numbers become available and as the communication link budget is better understood, that the output of the STK models will be significantly improved. At that time, an accurate simulation of the NPS-SCAT mission can be produced. The STK models and the MATLAB files can be used and upgraded for any future studies.

THIS PAGE INTENTIONALLY LEFT BLANK

# LIST OF REFERENCES


[1] AGI STK 8.1 Help

[2] F. Roßberg, “Structural design of a NPS CubeSat launcher”, Naval Postgraduate School, Monterey, CA, January 2008

[3] “Rocket & Space technology, Orbital Mechanics”, last reviewed February 2008

http://www.braeunig.us/space/orbmech.htm

[4] United Launch Alliance, “Atlas Launch System Mission Planner’s Guide”, Revision 10, January 2007

[5] AGI Software Technology Guide, last reviewed March 2008

http://www.agi.com/downloads/support/productSupport/literature/pdfs/CorporateBrochures/0307_AGI_Tech_Ref_Guide.pdf

[6] Howard D. Curtis, “Orbital Mechanics For Engineering Students”, Elsevier Butterworth-Heinemann, 2005

[7] NASA Johnson Space Center, Orbital Debris Program Office, “Orbital Debris Education Package”, last reviewed March 2008 http://orbitaldebris.jsc.nasa.gov/library/EducationPackage.pdf

[8] Gunter’s Space Page, Information on Launch vehicles, Satellites, Space Shuttle and Astronautic, last reviewed April 2008

http://space.skyrocket.de/index_frame.htm?http://skyrocket.de/space/doc_lau/atlas-5.htm

THIS PAGE INTENTIONALLY LEFT BLANK

# APPENDIX A. NPS-SCAT MODEL FILE DOCUMENTATION

FILE: NPS-SCAT.mdl

| Line | | Name | Description |
|---|---|---|---|
| Beginning | End | | |
| **1** | **504** | **NPS-SCAT model file** | |
| 1 | 11 | Description | |
| 12 | 17 | Cell efficiency | Definition of the efficiency of the solar cells (percent) |
| 18 | 35 | Main structure | Basic shape: four-sided cylinder |
| 36 | 51 | SunSensor | Measures the sun angle |
| 52 | 66 | Antenna | |
| 67 | 81 | Connectors | |
| 82 | 121 | Panels for SolarCells | Background for the solar cells |
| 82 | 96 | Panel | Panel for the sides |
| 97 | 108 | Panel2 | Panel for connector side |
| 109 | 121 | Panel3 | Panel for top and bottom |
| 122 | 137 | SolarCells | Triangular solar cells (group name npsscat) |
| 138 | 153 | Cellpair | Joining two solar cells |
| 154 | 168 | Experimental Cells | Triple-junction cells |
| 169 | 184 | TJC Cellpair | Joining two triple-junction cells |
| 185 | 209 | String assembly | Joining four solar cells (CellPanel) |
| 210 | 243 | Assembly SolarCell sidepanel | Joining six CellPanels on the Panel background |
| 244 | 298 | Top panel | Joining the sun sensor and solar cells |
| 299 | 353 | Bottom panel | Joining the sun sensor and solar cells |
| 354 | 391 | AntennaPanel | Joining the antenna with connectors and the solar cells |
| 392 | 407 | Spacer | Spacer on top and bottom of the CubeSat |
| 408 | 421 | Articulation Antenna | Definition of the movable antenna |
| 422 | 504 | NPS-SCAT ASSEMBLY | Assembly of all components |

# APPENDIX B. LAUNCH VEHICLE MODEL FILE DOCUMENTATION

FILE: atlas-v-heavy-NPS-SCAT.mdl

| Line | | Name | Description |
|---|---|---|---|
| Beginning | End | | |
| **1** | **24311** | **Atlas V heavy model file** | Created by STK, 2002 |
| **24312** | **26350** | **CubeSat launch vehicle** | |
| *24327* | *24485* | *D-advanced structure* | *NPS CubeSat Launcher* |
| 24334 | 24379 | Frontpart | |
| 24381 | 24419 | Bottompart | |
| 24421 | 24461 | Sidepart | |
| 24464 | 24485 | Assembley D-advanced structure | |
| *24489* | *24970* | *P-POD* | *5U P-POD by Cal Poly* |
| 24497 | 24523 | P-Cover | |
| 24526 | 24568 | Bracket | |
| 24571 | 24634 | Opening Mechanism | |
| 24637 | 24766 | Lid | |
| 24769 | 24970 | P-POD assembly | |
| *24975* | *25033* | *Baseplate and Lightband* | *Connection between ESPA-ring and structure (deployable)* |
| 24982 | 25000 | Baseplate | |
| 25003 | 25033 | Lightband | |
| *25037* | *25077* | *Electronics box* | |
| *25081* | *25244* | *ESPA-ring and Secondary payloads* | |
| 25088 | 25115 | ESPA-ring | |
| 25118 | 25161 | Secondary Payloads | |
| 25164 | 25186 | Lower Lightband | |

| | | | |
|---|---|---|---|
| 25189 | 25244 | ESPA-ring Assembly | |
| *25247* | *25315* | *NPSCuL-structure assembly* | |
| *25319* | *25729* | *Primary payload* | |
| 25326 | 25336 | PPLCone | Primary Payload cone - Connector between Payloads and ESPA-ring |
| 25338 | 25729 | NPSAT1 | Created by Dan Sakoda, SSAG, 2004 |
| **25732** | **26233** | **NPS-SCAT** | |
| *26234* | *26350* | *Scenario assembly* | |
| 26242 | 26266 | Launch Vehicle | Assembly PPLCone, ESPA-ring, NPSAT1 and ATLAS V |
| 26267 | 26275 | Deployable11 | Definition of separation between launch vehicle and NPSCuL |
| 26276 | 26287 | ScenarioLV | Assembly NPSCuL and Deployable11 |
| 26288 | 26297 | Deployable12 | Definition of separation between CubeSat and NPSCuL |
| 26298 | 26307 | Deployable14 | Definition of NPS-SCAT translation |
| 26308 | 26317 | Deployable15 | Definition of NPS-SCAT translation |
| 26318 | 26327 | Deployable16 | Definition of NPS-SCAT translation |
| 26328 | 26350 | ScenarioCubeSat | Assembly of the four CubeSat and the launch vehicle |

# APPENDIX C. DELTA-V CALCULATION

| BRIEF Description | | | |
|---|---|---|---|
| Inclination | 40 | | |
| launch azimuth | 60 | degree | |
| muE | 398600 | km^3/s^2 | |
| g0 | 9.80665 | m/sec^2 | |
| Isp | 450.5 | s | |
| Thurst | 99200 | N | |
| Initial mass | 36176 | kg | |
| | | | |
| ***Orbit 1 (after launch sequence)*** | | | |
| ra1 apogee | 6522.47 | km | |
| rp1 perigee | 6035.44 | km | |
| | | | |
| sma1 | 6278.955 | km | =(ra1+rp1)/2 |
| e | 0.03878 | | =(ra1-rp1)/(ra1+rp1) |
| va1 | 7.66432 | km/s | =SQRT(2*muE*(1/ra1-1/(2*sma1))) |
| | | | |
| ***Orbit 2*** | | | |
| rp2 | 6522.47 | km | |
| ra2 | 6728.14 | km | |
| | | | |
| sma2 | 6625.305 | km | =(ra2+rp2)/2 |
| e | -0.01552 | | =(rp2-ra2)/(ra2+rp2) |
| vp2 | 7.87784 | km/s | =SQRT(2*muE*(1/rp2-1/(2*sma2))) |
| va2 | 7.63703 | km/s | =SQRT(2*muE*(1/ra2-1/(2*sma2))) |
| | | | |
| **DeltaV-12** | **0.21352** | km/s | =vp2-va1 |
| | | | |
| ***Orbit 3*** | | | |
| ra3 | 6728.14 | km | =ra2 |
| rp3 | 6728.14 | km | =ra2 |
| | | | |
| sma3 | 6728.14 | km | =(ra3+rp3)/2 |
| e | 0 | | =(rp3-ra3)/(ra3+rp3) |
| vcirc3 | 7.69699 | km/s | =SQRT(2*muE*(1/ra3-1/(2*sma3))) |
| | | | |
| **DeltaV-23** | **0.05997** | km/s | = vcirc3-va2 |
| | | | |
| Finalmass-12 | 34469 | kg | =Initial_mass / (EXP(DeltaV_12/(Isp*g0*0.001))) |
| **Burntime-12** | **76.02** | s | =(Initial_mass-Finalmass_12)/Thurst*g0*Isp |
| Finalmass-23 | 34004 | kg | =Finalmass_12/(EXP(DeltaV_23/(Isp*g0*0.001))) |
| **Burntime-23** | **20.70** | s | =(Finalmass_12-Finalmass_23)/Thurst*g0*Isp |

# APPENDIX D. POWER BUDGET (MATLAB FILE)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%     2008 COPYRIGHT (C)   FELIX ROSSBERG     %%
%%          NAVAL POSTGRADUATE SCHOOL          %%
%%         SPACE SYSTEMS ACADEMIC GROUP        %%
%%                MONTEREY, CA                 %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%=============================================%
%              THE NPS-SCAT CUBESAT             %
%=============================================%

%%% Power budget NPS-SCAT (3 days of the scenario)

close all
clc
clear

% Power report is given in a 60 seconds time step

P = csvread('power_npsscat.csv',1,0);
[n,m]=size(P);
N=n/2;

L = csvread('link-budget.csv',1,1);
[M,n]=size(L);

for i=1:N
    link(i)=0;
end

for i=1:M
    for j=L(i,1):L(i,2)
        link(j)=1;
    end
end

%%%%% LOADS

% mpb - main processor board
%     power down, waiting for comm data   0.01
mpb_sleep = 0.01;
%     power up, comm data   0.1
mpb_on = 0.1;

% com - communication board
%     power down   0.01
com_sleep = 0.01;
%     transmit data   2.0
com_on = 2.0;

% sms - sms board
%     power down   0
sms_sleep = 0;
```

```
%     taking data   0.15
sms_on = 0.15;

for i=1:N
    if link(i)~=0
        mpb(i)=mpb_on;
        com(i)=com_on;
     else
        mpb(i)=mpb_sleep;
        com(i)=com_sleep;
    end
    if P(i,3)==1
        sms(i)=sms_on;
    else
        sms(i)=sms_sleep;
    end
    load(i)=mpb(i)+com(i)+sms(i);
    excess_power(i)=P(i,2);
end

for i=1:N
    excess_energy(i)=excess_power(i)/60;
end

%%%%% capacity of the battery
%      1.25Ah * 8.4V = 10.5 Wh
cap_batt_start = 10.5;

%%%%% conversion efficiency = 80%
eff=0.8;

%%%%% battery starts with 50% SOC
cap_batt(1) = cap_batt_start*0.5;

for i=2:N
    cap_batt(i) = cap_batt(i-1)+eff*excess_energy(i)-load(i)/60;
    if cap_batt(i)>cap_batt_start
        cap_batt(i)=cap_batt_start;
    end
    cap_batt_percent(i-1) = (cap_batt(i-1)/cap_batt_start)*100;
end

%%%%% plots

% Create figure
figure1 = figure('Name','NPS-SCAT Power Budget','Color',[1 1 1]);

subplot1 = subplot(2,1,1,'Parent',figure1,'YGrid','on','XGrid','on');
%xlim([80 240]);
box('on');
hold('all');
plot(load,'DisplayName','load','Parent',subplot1,'Color',[1 0 0]);
xlabel('Time [min]');
ylabel('Loads [W]');
```

```
subplot2 = subplot(2,1,2,'Parent',figure1,'YGrid','on','XGrid','on');
%xlim([80 240]);
box('on');
hold('all');
plot(cap_batt,'DisplayName','cap_batt','Parent',subplot2);
xlabel('Time [min]');
ylabel('Baterry capacity [Wh]');

figure2 = figure('Name','NPS-SCAT Baterry SOC','Color',[1 1 1]);

%plot3 = plot('Parent',figure2,'YGrid','on','XGrid','on');
box('on');
hold('all');
grid('on');
plot(cap_batt_percent,'DisplayName','cap_batt_percent',...
    'Color',[0 1 0]);
xlabel('Time [min]');
ylabel('Battery state of charge [%]');
```

# APPENDIX E. POWER BUDGET COMPARISON (MATLAB FILE)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%     2008 COPYRIGHT (C)   FELIX ROSSBERG     %%
%%          NAVAL POSTGRADUATE SCHOOL          %%
%%         SPACE SYSTEMS ACADEMIC GROUP        %%
%%                MONTEREY, CA                 %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%=============================================%
%              THE NPS-SCAT CUBESAT            %
%=============================================%

%%% Power budget NPS-SCAT comparison

close all
clc
clear

% Power report is given in a 60 seconds time step

P1 = csvread('power_011.csv',1,0);
P2 = csvread('power_100.csv',1,0);
P3 = csvread('power_001.csv',1,0);

[n,m]=size(P2);
N=n/2;

L1 = csvread('link_011.csv',1,1);
L2 = csvread('link_100.csv',1,1);
L3 = csvread('link_001.csv',1,1);

[M1,n]=size(L1);
[M2,n]=size(L2);
[M3,n]=size(L3);

for i=1:N
    link1(i)=0;
    link2(i)=0;
    link3(i)=0;
end

% only every third com-access is used

for i=1:M1
    for j=L1(i,1):L1(i,2)
        link1(j)=1;
    end
end

for i=1:M2
    for j=L2(i,1):L2(i,2)
        link2(j)=1;
    end
end
```

```
for i=1:M3
    for j=L3(i,1):L3(i,2)
        link3(j)=1;
    end
end

%%%%% LOADS

% mpb - main processor board
%     power down, waiting for comm data   0.01
mpb_sleep = 0.01;
%     power up, comm data   0.1
mpb_on = 0.1;

% com - communication board
%     power down   0.01
com_sleep = 0.01;
%     transmit data   2.0
com_on = 2.0;

% sms - sms board
%     power down   0
sms_sleep = 0;
%     taking data   0.15
sms_on = 0.15;

for i=1:N

    if link1(i)~=0
        mpb1(i)=mpb_on;
        com1(i)=com_on;
     else
        mpb1(i)=mpb_sleep;
        com1(i)=com_sleep;
    end
    if P1(i,3)==1
        sms1(i)=sms_on;
    else
        sms1(i)=sms_sleep;
    end

    load1(i)=mpb1(i)+com1(i)+sms1(i);
    excess_power1(i)=P1(i,2)-load1(i);

    if link2(i)~=0
        mpb2(i)=mpb_on;
        com2(i)=com_on;
     else
        mpb2(i)=mpb_sleep;
        com2(i)=com_sleep;
    end
    if P2(i,3)==1
        sms2(i)=sms_on;
    else
```

```
        sms2(i)=sms_sleep;
    end

    load2(i)=mpb2(i)+com2(i)+sms2(i);
    excess_power2(i)=P2(i,2)-load2(i);

    if link3(i)~=0
        mpb3(i)=mpb_on;
        com3(i)=com_on;
     else
        mpb3(i)=mpb_sleep;
        com3(i)=com_sleep;
    end
    if P3(i,3)==1
        sms3(i)=sms_on;
    else
        sms3(i)=sms_sleep;
    end

    load3(i)=mpb3(i)+com3(i)+sms3(i);
    excess_power3(i)=P3(i,2)-load3(i);

end

for i=2:N
    excess_energy1(i)=(excess_power1(i))*1/60;
    excess_energy2(i)=(excess_power2(i))*1/60;
    excess_energy3(i)=(excess_power3(i))*1/60;
end

%%%%% capacity of the battery
%      1.25Ah * 8.4V = 10.5 Wh
cap_batt_start = 10.5;

%%%%% conversion efficiency = 80%
eff=0.8;

%%%%% SOC at beginning of simulation
SOC_start=0.5;

cap_batt1(1) = cap_batt_start*SOC_start;
cap_batt2(1) = cap_batt_start*SOC_start;
cap_batt3(1) = cap_batt_start*SOC_start;

for i=2:N
    cap_batt1(i) = cap_batt1(i-1)+eff*excess_energy1(i);
    if cap_batt1(i)>cap_batt_start
        cap_batt1(i)=cap_batt_start;
    end
    cap_batt_percent1(i-1) = (cap_batt1(i-1)/cap_batt_start)*100;

    cap_batt2(i) = cap_batt2(i-1)+eff*excess_energy2(i);
    if cap_batt2(i)>cap_batt_start
        cap_batt2(i)=cap_batt_start;
    end
```

```
    cap_batt_percent2(i-1) = (cap_batt2(i-1)/cap_batt_start)*100;

    cap_batt3(i) = cap_batt3(i-1)+eff*excess_energy3(i);
    if cap_batt3(i)>cap_batt_start
        cap_batt3(i)=cap_batt_start;
    end
    cap_batt_percent3(i-1) = (cap_batt3(i-1)/cap_batt_start)*100;

end

%%%%% plots

% Create figure
figure1 = figure('Name','NPS-SCAT Power Budget');

% Create subplot
subplot(2,1,1,'Parent',figure1);
xlim([0 7200]);
box('on');
grid('on');
hold('all');

% Create multiple lines using matrix input to plot
plot1(1) = plot(cap_batt_percent1);
plot1(2) = plot(cap_batt_percent2);
plot1(3) = plot(cap_batt_percent3);
set(plot1(1),'DisplayName','cap_batt_percent1');
set(plot1(2),'DisplayName','cap_batt_percent2');
set(plot1(3),'DisplayName','cap_batt_percent3');
xlabel('Time [min]');
ylabel('Battery state-of-charge [%]');

% Create subplot
subplot(2,1,2,'Parent',figure1);
xlim([0 7200]);
% ylim([0 3]);
box('on');
grid('on');
hold('all');

plot2(1) = plot(load1);
plot2(2)= plot(load2);
plot2(3) = plot(load3);
set(plot2(1),'DisplayName','load1');
set(plot2(2),'DisplayName','load2');
set(plot2(3),'DisplayName','load3');
xlabel('Time [min]');
ylabel('Loads [W]');
```

THIS PAGE INTENTIONALLY LEFT BLANK

# INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   8725 John J. Kingman Rd., STE 0944
   Ft. Belvoir, VA 22060-6218

2. Dudley Knox Library, Code 52
   Naval Postgraduate School
   Monterey, CA

3. Space Systems Academic Group
   Naval Postgraduate School
   Monterey, CA

4. Helmut–Schmidt–Universität – Universität der Bundeswehr Hamburg
   Prüfungsamt Fakultät für Maschinenbau

5. Professur für Mess- und Informationstechnik
   Helmut–Schmidt–Universität – Universität der Bundeswehr Hamburg
   Hamburg, Germany

6. NASA Visiting Professor
   James H. Newman
   Naval Postgraduate School
   Monterey, CA